N° d'Ordre : 2

THÈSES

présentées

à la Faculté des Sciences de l'Université de Nantes

pour obtenir

le Grade d'Ingénieur-Docteur

par

Serge AOUSTIN

1re THÈSE : Étude de méthodes numériques directes pour la résolution de systèmes d'équations linéaires sur I.B.M. 650 Standard.

2e THÈSE : Propositions données par la Faculté.

Soutenues le 25 Mars 1963 devant la Commission d'Examen

MM. BRILLOUET, *Président*
PAUC
PIRONNEAU } *Examinateurs*
LACORD

N° d'ordre : 2

THESES

Présentées

A LA FACULTE DES SCIENCES de L'UNIVERSITE DE NANTES

pour obtenir

le grade d'INGENIEUR-DOCTEUR

par

Serge AOUSTIN

1ère Thèse : Etude de méthodes numériques directes pour la résolution de systèmes d'équations linéaires sur I.B.M. 650 Standard.

2ème Thèse : Propositions données par la Faculté

Soutenues le 25 mars 1963 devant la Commission d'Examen

MM. BRILLOUET *Président*

PAUC
PIRONNEAU *Examinateurs*
LACORD

R. UNIV.
BIBLIOTHEEK
LEIDEN

UNIVERSITE de NANTES

FACULTE des SCIENCES

Doyen ..... M. BRILLOUET

Assesseur ..... M. PIRONNEAU

Professeurs Titulaires	MM. BRILLOUET	Mathématiques
	PAUC	"
	PIRONNEAU	Mécanique
	RAVILLY	"
	BOULLOUD	Physique
	COLLET	"
	DUNOYER	Chimie
	GRAFF	"
	VAUGIEN	Biologie Animale
	GARRIGUES	Biologie Végétale
	GOURINARD	Géologie

Professeurs et Maîtres de Conférences	MM.	MOLINARO	Mathématiques
		PETRESCU	"
		VIVIER	"
		BALQUET	Mécanique
		AVIGNON	Physique
		GOUREAUX	"
		LASKAR	"
		LERMUSIAUX	"
		MEZENCEV	"
		MINN	"
		MOUTON	"
		JOLIVET	Chimie
		GOBIN	"
		SELEGNY	"
		RIO	"
		WOJTKOWIAK	"
		PIERI	Biologie Animale
		SELLIER	"
	Mlle	DEBRAUX	Biologie Végétale

Secrétaire ..... M. PAILLET

Digitized by the Internet Archive
in 2026

https://archive.org/details/IA41549214_0001

La présente étude a été effectuée au Centre de Calcul de l'Ecole Nationale Supérieure de Mécanique de Nantes et menée à bien sous la direction de Monsieur le Professeur BRILLOUET, Doyen de la Faculté des Sciences de Nantes, Directeur du Centre de Calcul.

Je tiens à exprimer toute ma gratitude à Monsieur le Professeur BRILLOUET pour l'intérêt permanent qu'il a manifesté vis à vis de mon travail et pour les conseils éclairés qu'il n'a cessé de me prodiguer.

Ma reconnaissance s'adresse également à Messieurs les Professeurs qui ont bien voulu me faire l'honneur de composer le Jury.

Je remercie vivement tous mes camarades du Centre de Calcul de l'E.N.S.M. ainsi que les personnes qui ont participé à la bonne présentation et à l'impression de ce fascicule.

TABLE DES MATIERES

Introduction 1

Première partie : Théorie des diverses méthodes 5

Chapitre 1 METHODES GENERALES 7

1.1. Méthode de JORDAN 7
1.1.1. Exposé de la méthode 7
1.1.2. Programmation 10
1.1.3. Nombre d'opérations élémentaires 12
1.1.4. Résultats 13

1.2. Méthode de GAUSS 20
1.2.1 Principe 20
1.2.2. Programmation 20
1.2.3. Nombre d'opérations élémentaires 22
1.2.4. Résultats 23

1.3. Amélioration de la méthode de JORDAN - Méthode de JORDAN - Itérations 26
1.3.1. Principe 26
1.3.2. Programmation 26
1.3.3. Nombre d'opérations 28
1.3.4. Résultats 29

1.4. Méthode des GRADIENTS CONJUGUES 36
1.4.1 Principe 36
1.4.2 Programmation 38
1.4.3. Nombre d'opérations élémentaires 39
1.4.4. Résultats 41

Chapitre 2 METHODES PARTICULIERES 47

2.1. Méthode de JORDAN Progressive 47
2.1.1. Exposé théorique 47
2.1.2. Nombre d'opérations élémentaires 49
2.1.3. Programmation 50
2.1.4. Application à la résolution des systèmes linéaires à MATRICE BANDE 52
2.1.5. Résultats 55
2.1.5.1. 55
2.1.5.2. Cas des systèmes linéaires à MATRICE BANDE 57

2.2. Méthode de découpage de la matrice 59
2.2.1 Principe 59
2.2.2. Programmation 60
2.2.3. Résultats 65

Chapitre 3 REMARQUES GENERALES 69

3.1. Comparaison des méthodes de GAUSS et de JORDAN 69

3.2. Influence de l'arrondi du résultat de chaque opération arithmétique en virgule flottante sur les résultats définitifs 69

3.3. Systèmes linéaires mal conditionnés - Cas particulier des matrices de HILBERT 71

3.4. Pertes de précision détectées au cours de la résolution d'un système linéaire 71

Conclusion 75

Deuxième partie : Annexes 79

Annexe 1 : *Programmes pour l'ordinateur I.B.M. 650 Standard*

I,1. Sous-Programmes de Résolution de Systèmes linéaires en SIMPLE PRECISION et en DOUBLE PRECISION. Méthode de JORDAN (Références : JSP1 et JDP1) 83
I,1.1. Notices d'utilisation 83
I,1.2. Note explicative du Sous-Programme en Simple Précision 87

I,2. Sous-Programmes de Résolution de Systèmes linéaires et d'Inversion de Matrice en SIMPLE PRECISION et en double PRECISION. Méthode de JORDAN (Références JSP4 et JDP4) 101
I,2.1. Notices d'utilisation 101
I,2.2. Note explicative du Sous-Programme en SIMPLE PRECISION 104

I,3. Sous-Programmes de Résolution de systèmes linéaires en SIMPLE PRECISION et en DOUBLE PRECISION. Méthode de GAUSS (Références GSP1 et GDP1) 124
I,3.1. Notices d'utilisation 124
I,3.2. Note explicative du sous-Programme en SIMPLE PRECISION 127

I,4. Sous-Programmes de Résolution de Systèmes linéaires en SIMPLE PRECISION. Méthode de JORDAN Itérations (Référence JSP) 146
I,4.1. Notice d'utilisation 146
I,4.2. Note explicative 148

I,5. Programmes de Résolution de Systèmes linéaires en SIMPLE PRECISION et en DOUBLE PRECISION. Méthode de JORDAN Progressive (Références JSP2 et JDP2) 179
I,5.1. Notices d'utilisation 179
I,5.2. Note explicative du Programme en SIMPLE PRECISION 184

I,6. Sous-Programmes de Résolution de Systèmes linéaires à MATRICE-BANDE en SIMPLE PRECISION et en DOUBLE PRECISION. Méthode de JORDAN Progressive (Références JSP3 et JDP3) 200
I,6.1. Notice d'utilisation 200
I,6.2. Note explicative du Sous-Programme en SIMPLE PRECISION 205

Annexe II : *Systèmes linéaires* 207

Annexe III : *Bibliographie* .....

INTRODUCTION

L'importance de la résolution des systèmes linéaires est admise depuis longtemps. La formulation mathématique de très nombreux problèmes conduit en effet à de tels systèmes. Nous citerons seulement comme exemples : le lissage des courbes expérimentales par la méthode des moindres carrés et les problèmes aux limites des équations différentielles ou équations aux dérivées partielles.

Les méthodes de résolution des systèmes linéaires existent en très grand nombre. Et plusieurs tentatives de classification ont été faites. Deux grandes classes de méthodes sont généralement admises :

- les méthodes directes dont le type est la méthode des déterminants
- et les méthodes indirectes ou itératives.

Si nous nous référons à la classification publiée par George E. FORSYTHE [Annexe III, (11)], le nombre de méthodes existantes, plus ou moins différentes, est environ égal à 90 .

Nous nous sommes limités ici à l'étude des principales méthodes directes appliquées à la résolution de systèmes linéaires quelconques, et à celle de la méthode des *Gradients conjugués* que l'on peut indifféremment considérer comme méthode directe ou indirecte. Il n'est pas moins vrai que si les méthodes indirectes sont peu efficaces dans le cas général, elles peuvent être préférables à toutes les méthodes directes dans certains cas particuliers.

Parallèlement à l'étude de quelques méthodes directes, nous avons considéré plus particulièrement trois problèmes importants :

- la résolution des systèmes linéaires mal conditionnés.
  Ils peuvent être détectés à l' aide de la perte maximale de précision rencontrée au cours d'une première résolution. Les résultats obtenus seront nettement améliorés en calculant avec un nombre suffisant de chiffres significatifs ou encore en utilisant la méthode de JORDAN - Itérations .
- la résolution des systèmes d'ordre élevé.
  A cet effet, nous avons considéré la méthode du Découpage de la Matrice et celle de JORDAN - Progressive.
- finalement, la résolution des systèmes linéaires à MATRICE-BANDE . Ce problème se présente dans l'intégration des équations différentielles avec conditions aux limites. Deux méthodes peuvent être utilisées : les GRADIENTS CONJUGUES et JORDAN - Progressive.

Tous ces problèmes traités sur Ordinateur, font l'objet de notre exposé qui s'organise de la façon suivante. Dans une première partie, nous trouvons l'étude détaillée des méthodes de résolution de systèmes linéaires que nous avons considérées. Chaque étude comprend : un exposé sommaire du principe de la méthode,

des considérations sur les divers modes de programmation possibles, et finalement la présentation des résultats obtenus à l'aide des programmes écrits. Dans une seconde partie (Annexes), nous exposons le détail des programmes jugés les meilleurs par les méthodes que nous avons retenues comme les plus intéressantes [Annexe I] , et nous fournissons les données des principaux systèmes linéaires que nous avons résolus [Annexe II] ainsi qu'une bibliographie sommaire [Annexe III] .

Tous nos essais ont été effectués sur l'Ordinateur I.B.M. 650 Standard du Centre de Calcul Scientifique de l'E.N.S.M. de Nantes. Aussi avons-nous effectué la plus grande partie de nos calculs en virgule flottante simulée. La virgule flottante utilisée est du type *Flair*. Elle est simulée en simple et en double précision à l'aide de Sous Programmes convenables écrits par le Centre, et dont nous ne considèrerons pas les détails.

NOTATIONS

Les notations particulières que nous utiliserons, seront définies au moment voulu. Nous allons seulement ici définir quelques notations générales.

Chaque système linéaire de n équations à n inconnues que nous aurons à résoudre, s'écrit

$$\sum_{j=1}^{n} a_{ij}\, x_j = b_i \qquad (i = 1, 2, 3, \ldots\ldots, n)$$

ou sous forme développée :

$$\begin{cases} a_{11} x_1 + a_{12} x_2 + a_{13} x_3 + \ldots\ldots + a_{1n} x_n = b_1 \\ a_{21} x_1 + a_{22} x_2 + a_{23} x_3 + \ldots\ldots + a_{2n} x_n = b_2 \\ \text{----------} \\ a_{n1} x_1 + a_{n2} x_2 + a_{n3} x_3 + \ldots\ldots + a_{nn} x_n = b_n \end{cases}$$

Avec les notations matricielles, nous poserons :

$$A = \begin{bmatrix} a_{11} & a_{12} & \ldots\ldots & a_{1n} \\ a_{21} & a_{22} & \ldots\ldots & a_{2n} \\ & & & \\ a_{n1} & a_{n2} & \ldots\ldots & a_{nn} \end{bmatrix} \qquad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

d'où l'écriture matricielle du système :

$$A\,x = b$$

Dans tout ce qui suit, nous supposerons D = dét. $A \neq 0$, ce qui assure l'existence et l'unicité de la solution du système.

Nous pouvons être amenés à résoudre une série de systèmes linéaires ayant tous la même matrice A, et des seconds membres différents. La résolution de tels systèmes peut se faire simultanément. Si nous désignons par «p» le nombre de séries de seconds membres traitées, nous aurons toujours la même écriture matricielle :

$$A\,x = b$$

mais avec des matrices x et b qui auront n lignes et p colonnes.

Dans le cas des méthodes opérant par élimination, nous conviendrons de représenter par :

$$a_{ij}^{k}$$

la valeur du coefficient a_{ij} après k éliminations. L'indice supérieur indique toujours le nombre d'éliminations déjà effectuées, lorsqu'il est effecté à une lettre minuscule.

Enfin nous désignerons toujours par N le nombre d'opérations arithmétiques nécessaires pour obtenir la solution d'un système linéaire.

PREMIERE PARTIE

THEORIE des DIVERSES METHODES

Chapitre 1

METHODES GENERALES

1 - 1 METHODE de JORDAN

1-1-1 EXPOSE de la METHODE

1-1-1-1- PRINCIPE

Nous résumerons le principe qui est élémentaire et bien connu. On élimine successivement les inconnues jusqu'à l'obtention d'une «matrice diagonale» ; c'est-à-dire que seuls les éléments de la diagonale principale sont différents de zéro. Le système final aura la forme :

$$\begin{aligned} a^n_{11}\, x_1 &= b^n_1 \\ a^n_{22}\, x_2 &= b^n_2 \\ a^n_{33}\, x_3 &= b^n_3 \\ &\cdots\cdots\cdots \\ a^n_{nn}\, x_n &= b^n_n \end{aligned}$$

dont la résolution est immédiate.

1-1-1-2 TECHNIQUE de l'ELIMINATION

La méthode pratique d'élimination est celle des «pivots», que nous exposerons sous l'une des formes possibles.

1-1-1-2-1 Définitions du pivot et de l'équation pivot

Considérons par exemple les i° et j° équations et proposons-nous d'éliminer x_k entre ces deux équations :

$$a_{i_1}\, x_1 + a_{i_2}\, x_2 + \ldots\ldots + a_{ik}\, x_k + \ldots\ldots + a_{i_n}\, x_n = b_i$$

$$a_{j_1}\, x_1 + a_{j_2}\, x_2 + \ldots\ldots + a_{jk}\, x_k + \ldots\ldots + a_{j_n}\, x_n = b_j$$

Multiplions ces deux équations par des coefficients m_i et m_j ; il faudra que :

$$m_i\, a_{ik} + m_j\, a_{jk} = 0$$

Si a_{ik} est différent de zéro, nous pourrons prendre :

$$m_j = 1 \qquad \text{et} \qquad m_i = -\frac{a_{jk}}{a_{ik}}$$

Le coefficient a_{ik} est alors appelé «PIVOT» et la i° équation, à laquelle il appartient, est «l'EQUATION PIVOT». Par conséquent, pour éliminer la k° inconnue dans la j° équation, nous lui ajoutons l'équation pivot multipliée par m_i. La j° équation prend la forme suivante, l'équation pivot restant inchangée :

$$a^1_{j_1}\, x_1 + a^1_{j_2}\, x_2 + \ldots\ldots + a^1_{j,k-1}\, x_{k-1} + 0 + a^1_{j,k+1}\, x_{k+1} + \ldots\ldots + a^1_{jn}\, x_n = b^1_j$$

Cette transformation sera appliquée à toutes les équations du système linéaire exceptée l'équation pivot, en faisant :

$$j = 1, 2, \ldots, (i-1), (i+1), \ldots, n$$

A la fin de ces opérations, la k° inconnue sera éliminée dans toutes les équations, sauf dans la i° . Le système linéaire se présente alors de la façon suivante si pour simplifier son écriture nous prenons $k = 1$ et $i = 1$. Le pivot est a_{11} et la première équation est l'équation pivot pour l'élimination de x_1 :

$$a_{11} x_1 + a_{12} x_2 + a_{13} x_3 + \ldots\ldots\ldots + a_{1n} x_n = b_1$$

$$a^1_{22} x_2 + a^1_{23} x_3 + \ldots\ldots\ldots + a^1_{2n} x_n = b^1_2$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

$$a^1_{n2} x_2 + a^1_{n3} x_3 + \ldots\ldots\ldots + a^1_{nn} x_n = b^1_n$$

Nous recommençons $(n - 1)$ fois les mêmes opérations sur les n équations. Mais nous ne choisirons jamais plus de pivot dans la première équation qui nous en a déjà fourni un. Finalement, nous obtiendrons le système diagonal dont il a été question plus haut.

1-1-1-2-2 Propriétés du pivot et de l'équation pivot

Comme conclusion pour le paragraphe précédent, nous pouvons énoncer les propriétés suivantes :

1°/ le pivot et l'équation pivot sont associés à l'élimination d'une même inconnue dans les $(n - 1)$ autres équations d'un système linéaire d'ordre n .

2°/ le pivot peut être choisi d'une façon quelconque pouvu que l'on rejette tout pivot :

a) nul

b) appartenant à une équation en ayant déjà fourni un.

Remarques

1°/ En utilisant les propriétés des opérations sur les lignes d'un déterminant, on démontre que le déterminant de la matrice d'un système linéaire d'ordre n est égal, *en valeur absolue*, au produit des n pivots utilisés pour le résoudre.

Par conséquent si à un stade quelconque de la résolution d'un système linéaire il est impossible de trouver un pivot non nul, cela signifie que l'on peut annuler tous les termes d'une colonne (ou d'une ligne) du déterminant de la matrice du système linéaire. Ce déterminant est donc nul et la matrice correspondante est singulière. La résolution numérique d'un tel système est impossible.

2°/ L'utilisation comme équation pivot d'une équation ayant déjà servi d'équation pivot aurait pour conséquence de réintroduire une inconnue déjà éliminée, dans les $(n - 1)$ autres équations du système linéaire.

1-1-1-2-3 Le choix du pivot

Mises à part les restrictions exposées au paragraphe précédent, les pivots peuvent être choisis d'une façon quelconque. Signalons néanmoins les méthodes de choix du pivot les plus couramment utilisées.

1-1-1-2-3-1 Choix du pivot sur la diagonale principale de la matrice du système linéaire

Les inconnues sont éliminées dans l'ordre où elles se présentent. De cette façon, la k° équation sera l'équation pivot pour l'élimination de la k° inconnue. Le pivot sera alors le terme situé à l'intersection

de la k° ligne et de la k° colonne de la matrice du système linéaire après les (k-1) premières éliminations.

Si on trouve un pivot nul, on permute l'équation correspondante avec une des équations n'ayant pas encore fourni de pivot.

Après les *n* éliminations, le système obtenu est à matrice diagonale.

1-1-1-2-3-2 Choix du pivot maximal dans une seule équation

Pour effectuer la première élimination, nous prenons la première équation comme équation pivot. Et nous éliminons l'inconnue ayant le coefficient le plus grand en valeur absolue (pivot maximal) dans cette équation. Nous opérons de la même façon pour les éliminations suivantes en utilisant successivement comme équations pivots : les 2°, 3°, ..., *n°* équations du système.

Si nous trouvons un pivot nul, la résolution numérique du système linéaire est impossible.

Après les *n* éliminations, il sera nécessaire d'effectuer des permutations de lignes pour faire apparaître un système linéaire à matrice diagonale.

1-1-1-2-3-3 Choix du pivot maximal dans toute la matrice

Considérons, comme au paragraphe *1-1-1-2-1*, l'élimination de la k° inconnue dans la j° équation avec la i° équation comme équation pivot, le coefficient a_{ik} supposé non nul étant le pivot. Pour cela, nous ajoutons à la j° équation : l'équation pivot multipliée par la quantité $m_i = -\frac{a_{jk}}{a_{ik}}$. Le calcul de ce multiplicateur donnera lieu en général à une erreur d'arrondi, donc à des coefficients erronés dans la nouvelle équation privée de x_k que l'on obtient. Ces coefficients erronés étant repris pour former de nouveaux coefficients, les erreurs d'arrondi se propageront. Pour limiter l'influence de ces erreurs, il est naturel de ne multiplier les équations que par des coefficients dont la valeur absolue soit inférieure à l'unité ; car ce faisant, l'erreur sur les coefficients des équations ne sera pas majorée par une telle multiplication.

Par conséquent, pour effectuer la première élimination, nous choisirons comme pivot : le coefficient maximum en valeur absolue dans la matrice du système. L'équation à laquelle il appartient sera l'équation pivot et l'inconnue dont il est le coefficient sera éliminée. Pour les éliminations suivantes, nous opérerons toujours de la même façon en tenant compte des restrictions imposées au choix du pivot (voir paragraphe *1-1-1-2-2-*) .

Le premier pivot nul entraîne la singularité de la matrice du système linéaire qui ne peut pas être résolu numériquement.

Une fois les *n* inconnues éliminées, il faudra permuter des lignes du système linéaire obtenu pour mettre en évidence la forme diagonale de sa matrice.

Remarque :

Bien que prenant le pivot maximum pour chaque inconnue, nous ne sommes pas sûrs de n'obtenir que des multiplicateurs m_i inférieurs à l'unité. En effet, lors de la k° élimination, les (k - 1) équations pivots des (k - 1) éliminations précédentes ne peuvent plus fournir de pivot. Et l'une au moins de ces équations pourra très bien avoir son coefficient de l'inconnue en cours d'élimination, supérieur au pivot.

1-1-1-3 APPLICATION au CALCUL de la MATRICE INVERSE

Si la matrice A est régulière, le système linéaire d'ordre n,

$$Ax = b$$

peut s'écrire

$$x = A^{-1} b$$

et en prenant pour b la matrice unité I d'ordre n , nous aurons :

$$x = A^{-1}$$

Par suite, le calcul de la matrice inverse A^{-1} revient à résoudre simultanément n systèmes linéaires ayant même matrice A et les vecteurs colonnes successifs de la matrice unité d'ordre n comme vecteurs seconds membres.

1-1-2 PROGRAMMATION

1-1-2-1 TECHNIQUES de l'ELIMINATION

Résoudre un système linéaire d'ordre n à l' aide de la méthode de JORDAN , consiste à effectuer n éliminations qui peuvent être conduites de différentes façons, mis à part le choix du pivot.

1-1-2-1-1-Première méthode d'élimination

Dans cette première méthode, nous conservons les équations pivots telles qu'elles se présentent, ce qui conduit à calculer pour chaque élimination : les $(n - 1)$ multiplicateurs. De cette façon, une fois que toutes les inconnues seront éliminées, nous diviserons les seconds membres par le pivot convenable pour obtenir les racines.

1-1-2-1-2 Deuxième méthode d'élimination

Contrairement à la méthode précédente, nous divisons chaque équation pivot par le pivot, ce qui nous apporte les avantages suivants :

- A chaque élimination, nous libérons n mémoires au lieu de $(n - 1)$ avec la première méthode. En effet, le pivot devenant égal à l'unité, il est inutile de le conserver en mémoire, de même que les $(n - 1)$ coefficients de l'inconnue éliminée qui sont alors nuls. Cet avantage sera exploité pour le calcul de la matrice inverse, comme nous le verrons plus loin.

- Le calcul des $(n - 1)$ multiplicateurs n'est plus nécessaire. Nous prenons directement, dans chaque équation, le coefficient de l'inconnue à éliminer.

- Nous disposons des racines, immédiatement après la fin de la n^{o} élimination.

Tous ces avantages se traduisent encore par une diminution du nombre des opérations arithmétiques nécessaires à l'obtention de la solution. La seconde méthode nous permet de gagner : $\frac{n(n-1)}{2}$ opérations arithmétiques.

1-1-2-2 CALCUL de la MATRICE INVERSE

Si nous utilisons la méthode de JORDAN pour calculer la matrice inverse d'une matrice A d'ordre n , nous sommes amenés à résoudre le système linéaire :

$$AX = I$$

La matrice second membre I est la matrice unité d'ordre n . La solution X n'est autre que la matrice inverse A^{-1} . Pour effectuer ce calcul, il est inutile de consacrer $n \times n$ mémoires à l'enregistrement de la matrice unité I. En effet, chaque colonne de cette matrice reste inchangée tant que le terme correspondant dans le second membre de l'équation pivot est nul ; c'est-à-dire, tant que l'équation de même rang que la colonne considérée de la matrice unité, n'a pas été prise comme«équation pivot» . Par conséquent, à chaque élimination, une nouvelle colonne de la matrice unité participe aux calculs. Initialement, ses n termes sont nuls sauf celui de même rang que l'équation pivot qui est égal à l'unité. Après l'élimination en cours, nous aurons n nouveaux termes qu'il faudra conserver en mémoires. Si nous utilisons la seconde méthode d'élimination signalée plus haut, nous libèrons n mémoires à la fin de chaque élimination. Nous les utiliserons immédiatement au stockage des nouveaux termes de la dernière colonne de I introduite. En opérant de cette façon, à la fin des n éliminations, nous avons calculé la matrice inverse que nous trouvons en mémoires à la place de la matrice A du système linéaire.

Remarque

Il est possible de faire le calcul d'une matrice inverse en utilisant la première méthode d'élimination, mais nous ne pouvons plus la substituer à la matrice initiale, comme précédemment.

1-1-2-3 ORGANISATION GENERALE du PROGRAMME

Lors de chaque élimination, si nous ne prenons pas le pivot sur la diagonale principale de la matrice du système linéaire, le programme de la méthode de JORDAN (avec ou sans le calcul de la matrice inverse) peut être construit de différentes façons.

1-1-2-3-1 Première méthode de programmation

Dans cette première méthode, nous séparons toutes les opérations. Chaque élimination commence alors par le choix du pivot soit dans une seule équation, soit dans toute la matrice. Puis nous effectuons les calculs d'élimination suivant l'une des deux méthodes citées plus haut. Finalement, une fois les n éliminations terminées, nous réordonnons les résultats. Ce dernier travail s'effectue par permutations de lignes dans le cas de la résolution d'un système linéaire ; et par permutations de lignes et decolonnes dans le cas d'une inversion de matrice.

1-1-2-3-2 Deuxième méthode de programmation

La méthode précédente peut être considérablement améliorée. C'est ce que nous avons fait d'une manière progressive pour aboutir finalement à ce que nous appellerons la seconde méthode de programmation qui procède de la façon suivante. Tout en effectuant les calculs de la k^{o} élimination, nous choisissons le pivot pour l'élimination suivante et nous réordonnons les résultats ; de sorte que chaque coefficient de la matrice et des seconds membres n'est appelé qu'une seule fois au cours de chaque élimination. Ceci est très important, car les instructions d'appel de coefficients ne peuvent pas être optimisées et sont des causes de pertes de temps. Le choix du pivot est basé sur le fait que tout résultat d'une élimination peut servir de pivot lors de l'élimination suivante à condition qu'il appartienne à une ligne et à une colonne pouvant fournir un pivot et n'en ayant pas encore fourni. Il est donc logique et préférable d'effectuer le choix de pivot dès qu'un tel terme se trouve disponible dans les accumulateurs, plutôt que d'aller le chercher ultérieurement dans sa mémoire. Seul le premier choix de pivot échappe à ce mode opératoire. IL pourra être fait en explorant tous les coefficients de la matrice du système supposée préalablement enregistrée, ou dans le cas contraire : pendant

l'enregistrement des données. Le réarrangement des résultats de chaque élimination consiste à reclasser convenablement :

- l'équation pivot parmi les équations ayant déjà fourni un pivot,
- ainsi que dans le cas de l'inversion de matrice, la dernière colonne introduite de la matrice unité, parmi les colonnes précédemment utilisées.

Cette seconde méthode de programmation nécessite le stockage de chaque équation pivot dans ($n + p$) mémoires de travail. Nous pourrons profiter de ce transfert pour diviser l'équation pivot par le pivot (2° méthode d'élimination). En opérant de cette dernière façon avec la seconde méthode de programmation. Ces résultats sont disponibles immédiatement après la fin de la n^o élimination. Au contraire, si nous utilisons la première méthode d'élimination, nous terminerons les calculs en divisant les seconds membres par les pivots convenables. Le lecteur trouvera en annexe [Annexe I, 1 et 2], tous les détails relatifs aux programmes de résolution de systèmes linéaires (avec ou sans inversion de matrice) à l'aide de la méthode de JORDAN et utilisant les deuxièmes méthodes de programmation et d'élimination.

1-1-2-4 METHODES de CALCUL

Toutes les méthodes de programmation, que nous venons de définir, peuvent calculer indifféremment en virgule flottante simple ou double précision et en virgule fixe. Il suffit d'utiliser le sous-programme de calcul approprié, la logique restant inchangée.

La virgule flottante simulée est du type FLAIR (exposant à gauche), ce qui permet d'effectuer la recherche du pivot maximal en virgule fixe.

Dans le cas de la virgule flottante en double précision, chaque coefficient occupe deux mémoires que l'on peut répartir de différentes façons. Nous avons choisi de passer de l'adresse de la partie gauche d'un nombre, à celle de la partie droite en ajoutant un facteur de translation S. En adoptant cette disposition, toutes les parties gauches des données occupent un premier bloc de mémoires et les parties droites : un second bloc identique au précédent et qui s'en déduit par une translation S. Cette disposition permet de conserver pour le programme en virgule flottante double précision, la même logique que pour le programme en simple précision. Elle présente en plus l'avantage de faciliter le passage de la simple à la double précision, ou inversement. Finalement, le programme en virgule flottante double précision s'obtient à partir de celui en simple précision, dans lequel :

- nous multiplions par deux les instructions d'appel et de stockage des données et des résultats.
- nous remplaçons l'arithmétique en virgule flottante simple précision par celle en double précision.

1-1-3 NOMBRE d'OPERATIONS ELEMENTAIRES

Nous désignerons par « n » l'ordre du système linéaire à résoudre, et par « p » le nombre de séries de seconds membres pour lesquelles nous désirons obtenir la solution. Les calculs d'élimination sont supposés être effectués suivant ce que nous avons appelé la «Deuxième méthode d'élimination».

1-1-3-1 NOUS NE CALCULONS PAS LA MATRICE INVERSE

Supposons que nous abordons la k^o élimination.

Nous divisons en premier lieu, l'équation pivot par le pivot. Ceci nécessite :

(n - k + p) divisions.

Ensuite, nous effectuons les calculs d'élimination sur les $(n-1)$ équations du système. Chaque équation se compose de : $(n-k+1+p)$ termes, avant la k° élimination, et de $(n-k+p)$ termes ensuite. Le calcul de chacun de ces termes nécessite : 2 opérations (une multiplication et une addition). Pour le calcul de chaque nouvelle équation, il faudra donc :

$$2(n-k+p) \text{ opérations}$$

et pour les $(n-1)$ nouvelles équations :

$$2(n-1)(n-k+p)$$

Par suite, la k° élimination nécessitera :

$$(2n-1)(n-k+p)$$

La résolution du système linéaire sera effective après n éliminations de ce genre ; d'où le nombre total d'opérations :

$$N = (2n-1) \sum_{k=1}^{n} (n-k+p)$$

ou

$$\boxed{N = \frac{1}{2} n(2n-1)(n+2p-1)}$$

1-1-3-2 NOUS CALCULONS LA MATRICE INVERSE

Supposons encore que nous abordions la k° élimination:

Nous divisons tout d'abord l'équation pivot par le pivot, avec introduction du terme d'une nouvelle colonne de la matrice inverse. Nous effectuerons :

$$(n+p) \text{ divisions.}$$

Ensuite, nous effectuons les calculs d'élimination sur les $(n-1)$ équations du système. Chaque équation se compose de $(n+p)$ termes avant la k° élimination et du même nombre de termes après, du fait de l'introduction d'une nouvelle colonne de la matrice inverse. Le calcul de chacun de ces termes nécessite : 2 opérations.

Pour le calcul de chaque nouvelle équation, il faudra donc :

$$2(n+p) \text{ opérations.}$$

et pour les $(n-1)$ nouvelles équations :

$$2(n-1)(n+p) \text{ opérations.}$$

Par suite, la k° élimination nécessitera :

$$(2n-1)(n+p) \text{ opérations.}$$

Le nombre d'opérations est indépendant du rang k de l'élimination. La résolution du système linéaire avec inversion de la matrice nécessite «n» éliminations. Au total, le nombre d'opérations sera donc :

$$\boxed{N = n(2n-1)(n+p)}$$

1-1-4 RESULTATS

1-1-4-1 PROGRAMMES en VIRGULE FLOTTANTE

En faisant le choix du pivot maximal dans toute la matrice, et en calculant en virgule flottante simple précision, nous avons essayé divers modes de programmation possibles à partir des méthodes d'élimination et de programmation énoncées au paragraphe précédent. Ayant constaté qu'avec tous ces programmes

les résultats obtenus étaient pratiquement identiques, nous avons étudié les temps de calcul et les possibilités maximales pour chaque programme. Nos essais ont vite montré que, si les programmes avaient des possibilités maximales comparables, le plus rapide était indiscutablement celui écrit à partir des deuxièmes méthodes d'élimination et de programmation. Ce programme permet un gain de temps de l'ordre de 37% par rapport au programme le plus lent qui utilise les premières méthodes d'élimination et de programmation. La méthode de choix du pivot maximal dans toute la matrice, qui consiste à effectuer le choix du pivot de la $(k+1)^e$ élimination tout en calculant les résultats de la k^e, permet à elle seule un gain de temps de 18% par rapport au même programme faisant le choix du pivot avant le début de chaque élimination. Cette constatation est très importante. La nouvelle méthode de choix du pivot permet d'atténuer considérablement la perte de temps que l'on peut reprocher au choix du pivot maximal dans toute la matrice par rapport : soit au choix du pivot dans une seule équation, soit au pivot pris obligatoirement sur la diagonale principale (nous éliminons alors les inconnues dans l'ordre où elles se présentent). Nous aurons d'ailleurs l'occasion de revenir sur ce problème, un peu plus loin.

Dans tout ce qui suit, chaque fois que l'on parlera de choix du pivot maximal dans toute la matrice, nous ne considèrerons plus que les programmes écrits à partir des deuxièmes méthodes d'élimination et de programmation. Ces programmes sont les plus rapides et se trouvent entièrement détaillés en annexe, [Annexe I, 1 et 2].

Nous allons maintenant donner pour ces programmes : trois tableaux indiquant les possibilités maximales et différents temps de calcul.

POSSIBILITES MAXIMALES	METHODE de JORDAN avec choix du pivot maximal dans toute la matrice, programmée en *VIRGULE FLOTTANTE*			
	Simple Précision		*Double Précision*	
	sans le calcul de la matrice inverse.	avec le calcul de la matrice inverse.	sans le calcul de la matrice inverse.	avec le calcul de la matrice inverse.
n_{MAX}	37	36	24	24
p_{MAX}	1	2	2	1

Remarque

La valeur p_{MAX} représente la valeur maximum permise pour p lorsque nous prenons n égal à n_{MAX}.

Ordre du système linéaire étudié ; nous avons toujours p = 1	METHODE de JORDAN avec choix du pivot maximal dans toute la matrice, programmée en *virgule flottante simple précision*			
	sans inversion de matrice		avec inversion de matrice	
	Temps de calcul	Nombre d'opérations	Temps de calcul	Nombre d'opérations
6	32 secondes	231	52 secondes	462
12	3 mn 28 s.	1794	5 mn 53 s.	3588
18	10 mn 52 s.	5985	18 mn 50 s.	11970
24	24 mn 40 s.	14 100	43 mn 40 s.	28 200
30	46 mn 42 s.	27 435		
36	72 mn 16 s.	47 286		

Ordre du système linéaire étudié ; nous avons toujours p = 1	METHODE de JORDAN avec choix du pivot maximal dans toute la matrice , programmée en *virgule flottante double précision*			
	sans inversion de matrice		avec inversion de matrice	
	Temps de calcul	Nombre d'opérations	Temps de calcul	Nombre d'opérations
6	1 mn 22 s.	231	2 mn 27 s.	462
12	9 mn 30 s.	1 794	17 mn 55 s.	3 588
18	30 mn 50 s.	5 985		11 970
24		14 100		28 200

Remarque 1

Si nous passons de la résolution d'un système linéaire avec une seule série de seconds membres ($p = 1$) et sans inversion de la matrice du système, au même problème avec inversion de matrice, le nombre d'opérations arithmétiques est multiplié par 2 et les temps de calcul par un coefficient légèrement inférieur à deux. L'écart que nous venons de constater, s'explique par le fait que le calcul supplémentaire de la matrice inverse n'augmente pas le nombre d'opérations logiques dans le même rapport que celui des opérations arithmétiques.

Remarque 2

Le passage, en virgule flottante, de la simple précision à la double précision conduit à multiplier tous les temps de calcul par un coefficient compris entre 2,5 et 3.

Remarque 3

Nous avons essayé les programmes précédents, en virgule flottante simple précision, sur un ordinateur I.B.M. 650 muni de la virgule flottante automatique. Cette dernière est environ 6 fois plus rapide que la virgule flottante simulée ; mais pour la résolution d'un système linéaire, on peut estimer que les calculs comptent seulement pour environ 60% du temps total en machine I.B.M. 650 Standard, et les gains de la virgule flottante automatique ne portent que sur cette partie. Si bien que d'une façon générale, les temps de calcul sont deux fois moins élevés. Il faut signaler aussi qu'en virgule flottante automatique, l'exposant du nombre se trouve à droite (positions 1 et 2 du mot), la comparaison des valeurs absolues de deux nombres ne peut donc pas se faire en virgule fixe comme avec les nombres en virgule flottante type FLAIR.

1-1-4-2 ESSAIS de PROGRAMMATION en VIRGULE FIXE

Nous avons programmé la méthode de JORDAN en virgule fixe en supposant une même position de virgule pour tous les coefficients de la matrice et des seconds membres du système linéaire à traiter. Cette position de virgule pourrá éventuellement varier au cours des calculs mais restera toujours commune à l'ensemble des coefficients gardés en mémoire. Tous les essais ont conduit à des temps de calcul supérieurs à ceux mesurés pour les programmes en virgule flottante ; ce qui peut, à première vue, sembler paradoxal. En fait, l'expérience montre qu'il n'est pas possible de programmer la méthode de JORDAN en virgule fixe, dans de bonnes conditions. Les opérations à effectuer ne s'y prêtent pas, surtout si nous voulons faire des calculs aussi précis que possible. En effet, que nous utilisions la première ou la seconde méthode d'élimination (voir au paragraphe précédent 1-1-2-1), nous rencontrerons toujours les deux mêmes types d'opérations :

a) Division. Il s'agit de diviser un nombre par un autre ayant le même cadrage (calcul d'un multiplicateur ou division de l'équation pivot par le pivot). Il faudra obligatoirement éviter les dépassements de capacité et si l'on veut une bonne précision : rechercher le nombre maximum de chiffres au quotient. Ce qui nous amène à effectuer les mêmes opérations que pour la division en virgule flottante.

b) Calcul d'élimination. Au cours de chaque élimination, les nouveaux coefficients des $(n - 1)$ nouvelles équations sont tous calculés de la même façon : à partir de l'ancien coefficient A et du coefficient correspondant de l'équation B suivant la formule $A + q B$. Le symbole « q » représente le multiplicateur convenable associé à chaque équation. Les coefficients A et B ont le même cadrage. Le multiplicateur « q » peut être, en valeur absolue, plus grand que l'unité dans le cas de la méthode de JORDAN. Si bien que le cadrage du résultat $A + q B$ pourra être différent de celui de A et B. Si, en valeur absolue, $A + q B$ est inférieur à

A ou B, il sera possible de le cadrer de la même façon que A et B. Cette opération sera immediate mais entraînera une perte de chiffres significatifs. Si au contraire A + q B est, en valeur absolue, supérieur à A ou B, il faudra aligner le cadrage de tous les nombres conservés en mémoire, sur celui de A + q B. Cette opération entraînera une augmentation du temps de calcul et une perte de chiffres significatifs sur tous les nombres en mémoires.

La programmation en virgule fixe nécessite donc de nombreux calculs de positions de virgule (ou d'exposants) pour cadrer chaque nouvelle quantité calculée, de la même façon que celles préalablement obtenues, et ceci en dehors de toute considération relative à la précision. Pour résoudre ce dernier problème, il faudrait en plus : chercher à calculer sur un nombre maximum de chiffres significatifs. Sans pouvoir espérer obtenir les mêmes résultats qu'avec la virgule flottante (nous avons toujours 8 chiffres significatifs), il faudrait cadrer tous les nombres de telle façon que le plus grand, en valeur absolue, n'ait aucun zéro à gauche dans sa mémoire. La réalisation de ce dernier impératif est possible, mais elle nécessite un programme compliqué utilisant un nombre prohibitif de mémoires de travail, et dont les temps de calcul sont alors nettement supérieurs à ceux des programmes en virgule flottante. Finalement la programmation en virgule fixe conduit à effectuer des opérations comparables à celles rencontrées en virgule flottante. On comprend donc qu'il était difficile d'espérer améliorer les temps de calcul par cette méthode. La meilleure précision des résultats est liée à la recherche du nombre maximum de chiffres significatifs, mais elle dépend encore des ordres de grandeurs relatifs des coefficients de la matrice et des seconds membres du système linéaire étudié.

1-1-4-3 INFLUENCE du MODE de CHOIX du PIVOT

En utilisant les programmes en virgule flottante simple précision, nous avons étudié l'influence du mode de choix du pivot sur les temps de calcul et sur la précision des résultats. En ce qui concerne les premiers, nous avons le tableau suivant :

Caractéristiques du système linéaire étudié : $p = 1$, $n =$	Méthode de JORDAN - Temps de calcul pour la résolution d'un système linéaire		
	Mode de choix du pivot		
	sur la diagonale principale	dans une seule ligne	dans toute la matrice
6	21 secondes	31 secondes	32 secondes
12	2 mn 32 s.	3 mn 15 s.	3 mn 28 s.
18	8 mn 18 s.	10 mn 19 s.	10 mn 52 s.

Par rapport au choix du pivot maximal dans toute la matrice : le choix du pivot dans une seule ligne permet un gain de temps d'environ 5 % , et l'élimination des inconnues dans l'ordre où elles se présentent, entraîne un gain de l'ordre de 30 %.

1-1-4-4 PRECISION des RESULTATS

Si pour le critère des temps de calcul, il est facile de détecter la meilleure méthode, il n'en va pas de même pour celui de la précision des résultats. Ceux-ci sont fonction de l'ordre relatif des équations quand on ne fait pas le choix du pivot dans toute la matrice. En considérant un système linéaire d'ordre 12 [Annexe II, 2] dont toutes les racines, connues exactement, sont en valeur absolue inférieures à 3,3 , nous avons relevé les écarts absolus suivants :

- 10^{-7} avec le choix du pivot dans toute la matrice, et ceci quel que soit l'ordre des équations.
- 3.10^{-7} avec le choix du pivot dans une seule ligne
- 6.10^{-5} en prenant le pivot sur la diagonale principale.

Pour les deux derniers cas, il s'agit d'écarts absolus maxima relevés en résolvant plusieurs fois le même système linéaire en modifiant à chaque fois l'ordre des équations.

Ces résultats ne nous permettent pas de conclure définitivement. Il semble néanmoins que le choix du pivot maximum dans toute la matrice conduise à une erreur absolue minimum sur les résultats, dans le cas de systèmes linéaires bien conditionnés. Cette erreur absolue porte généralement sur le dernier chiffre significatif fourni par la virgule flottante simple précision.

Dans le cas de systèmes linéaires instables ou mal conditionnés, les trois méthodes de choix du pivot conduisent toutes à des résultats imprécis qui ne fournissent qu'un ordre de grandeur des racines recherchées. Les résultats seront alors nettement améliorés par l'emploi de la méthode de JORDAN-Itérations (voir paragraphe *1-2*) ou tout simplement en effectuant les calculs en double précision. Avec cette dernière méthode, les temps de calcul sont de 2 à 3 fois plus importants, mais la précision des résultats est très satisfaisante. L'erreur porte généralement sur la partie droite des nombres en double précision (dix derniers chiffres significatifs).

A titre d'exemple, le tableau suivant donne les résultats obtenus avec un système linéaire d'ordre 6 [Annexe II, 5] fourni par une méthode de moindres carrés.

	RACINES		
	x_1	x_2	x_3
Valeurs exactes	8, 0000000	12 , 000000	17, 000000
Résultats obtenus avec la méthode de JORDAN en Simple Précision	8, 0912433	12, 001173	16, 773302
Résultats obtenus avec la méthode de JORDAN en Double Précision	8, 0000000	11, 999999	16, 999999
	000 573 2770	99999 41926	999 8723 285

	RACINES		
	x_4	x_5	x_6
Valeurs exactes	- 15, 000000	- 0, 30000000	- 30, 000000
Résultats obtenus avec la méthode de JORDAN en Simple Précision	- 14, 999244	- 0,30043796	- 29, 862350
Résultats obtenus avec la méthode de JORDAN en Double Précision	- 14, 999999	- 0, 29999999	- 29, 999999
	- 9999940993	- 9999719868	- 9999285969

Remarques

- L'unité en virgule flottante double précision s'écrit :

51 1000 0000 ⋮ 00 0000 0000

et nous l'écririons dans les tableaux précédents :

1,0000000

00 0000 0000

- Dans les deux méthodes de JORDAN utilisées, le choix du pivot s'est fait dans toute la matrice.

Comme toutes les autres méthodes, celle de JORDAN en double précision est inefficace dans le cas de systèmes linéaires dont les données sont entachées d'erreur. Par exemple, l'inversion d'une matrice de HILBERT dont les coefficients fractionnaires doivent être arrondis, fournit des résultats sans signification.

1-1-4-5 PERTE MAXIMUM DE PRECISION

A la fin de la résolution d'un système linéaire, en virgule flottante simple précision, par la méthode de JORDAN, nous disposons de la perte maximum de précision. Il s'agit du contenu d'une mémoire qui indique qu'au cours de la k° élimination, nous avons eu une perte de x chiffres significatifs. Ainsi, pour le système d'ordre 6 [Annexe II, 5] dont nous venons de donner les résultats: la perte maximum est de 5 chiffres (x = 5) et se produit au cours de la 2° élimination (k = 2) . Le calcul alors effectué est le suivant :

$$3,8547910 - \frac{4,5350120}{4,9448000} \times 4,2030760$$

$$- 3,8547565 = 0,00000345$$

Le nombre maximum de chiffres perdus et le moment précis de cette perte, orientent utilement le jugement des résultats.

1 - 2 METHODE de GAUSS

1-2-1 PRINCIPE

On élimine successivement les inconnues jusqu'à ce que chaque équation renferme une variable de moins que la précédente. Le but est donc d'obtenir un système dont la matrice est «triangulaire supérieure», c'est-à-dire que tous les éléments au dessous de la diagonale principale sont nuls, les termes de la diagonale principale étant tous égaux à l'unité , soit :

$$\begin{aligned} x_1 + a^1_{12} x_2 + a^1_{13} x_3 + \ldots\ldots\ldots\ldots + a^1_{1n} x_n &= b^1_1 \\ x_2 + a^2_{23} x_3 + \ldots\ldots\ldots\ldots + a^2_{2n} x_n &= b^2_2 \\ x_3 + \ldots\ldots\ldots\ldots + a^3_{3n} x_n &= b^3_3 \\ \text{----------} & \\ x_n &= b^n_n \end{aligned}$$

Un tel système se résout immédiatement par «substitution inverse» en tirant x_1 de la dernière équation ; puis en substituant la valeur obtenue dans l'équation précédente d'où l'on tire x_{n-1} ; et ainsi de suite jusqu'à la première équation qui donne x_1.

La technique de l'élimination ne diffère de celle utilisée pour la méthode de JORDAN, que sur les points suivants :

- chaque fois qu'une équation fournit un pivot, elle est mise de côté et ne participe pas aux calculs ultérieurs,
- on peut indifféremment :
  - prendre le pivot sur la diagonale principale de la matrice,
  - choisir le pivot maximal dans une seule équation ou dans toute la matrice.

Dans ce dernier cas, les multiplicateurs utilisés au cours de chaque élimination seront tous, en valeur absolue, inférieurs à l'unité.

1-2-2 PROGRAMMATION

Pour une étude complète de la programmation de la méthode de GAUSS avec choix du pivot maximal dans toute la matrice, nous renvoyons à la note explicative placée en [Annexe I, 3] . Nous nous contenterons ici de signaler les principaux problèmes.

1-2-2-1 ELIMINATIONS

Nous pouvons lors de chaque élimination, opérer de deux façons différentes. Soit que l'on divise l'équation pivot par le pivot avant de commencer les calculs d'élimination, soit que l'on utilise l'équation pivot telle qu'elle se présente. En calculant le nombre d'opérations arithmétiques correspondant à chaque cas, on constate qu'ils sont les mêmes quel que soit «p». Toutefois la première méthode facilite légèrement la programmation. C'est pourquoi nous l'avons choisie. Elle serait indispensable si l'on voulait calculer la matrice inverse du système en substituant à chaque élimination la dernière colonne introduite de la matrice inverse aux

coefficients de l'inconnue éliminée. Nous n'avons pas envisagé le problème d'inversion de matrice à l'aide de la méthode de GAUSS, car dans ce domaine elle ne présente aucun avantage sur la méthode de JORDAN.

L'appel répété des coefficients du système linéaire stockés en mémoire est une cause de perte de temps, les instructions correspondantes n'étant que très rarement optimisées. C'est pourquoi il faut réduire ces appels au strict minimum. Au cours de chaque élimination, chacun des coefficients doit participer à trois opérations :

- calcul de sa nouvelle valeur après l'élimination en cours.
- mise en mémoire convenable de ce nouveau coefficient.
- choix du pivot pour l'élimination suivante.

Ces trois opérations peuvent être effectuées en n'appelant qu'une seule fois le coefficient intéressé pour calculer sa nouvelle valeur que nous stockons immédiatement dans une mémoire convenable. Ceci ne pose pas de problème, car pour la méthode de GAUSS, il s'agit de bloquer les uns à la suite des autres, les nouveaux coefficients qui participeront à la suite des calculs. Tout résultat d'une élimination est susceptible d'être choisi comme pivot pour l'élimination suivante. On effectuera donc la recherche de ce pivot immédiatement après le calcul de chaque nouveau coefficient, alors qu'il se trouve encore dans les accumulateurs. Il suffira de prévoir la suppression du choix de pivot sur les coefficients correspondant aux seconds membres. Il n'y a que le choix du premier pivot qui ne pourra pas être fait exactement de la même façon, il sera réalisé en utilisant le programme d'élimination dans lequel on supprime momentanément tout calcul et stockage de résultat, en ne laissant subsister que l'appel des coefficients. Cette façon d'opérer permet un gain de temps considérable ; ce gain de temps a été chiffré pour la méthode de JORDAN; dans le cas de la méthode de GAUSS, on peut considérer qu'il est du même ordre de grandeur.

1-2-2-2 SUBSTITUTION INVERSE

Autre problème, propre à la méthode de GAUSS : la substitution inverse. C'est à ce stade que se pose réellement la mise en ordre des résultats. A cet effet, nous disposons de «*n*» mémoires associées aux «*n*» équations pivots. Chacune de ces mémoires contient des indications sur le rang de l'inconnue qui sera calculée à partir de l'équation pivot correspondante. Donc à chaque étape de la substitution inverse, nous connaîtrons le rang de la dernière inconnue calculée, ce qui nous permettra de la classer parmi les autres inconnues préalablement calculées. Ce problème est lié au choix du pivot maximal , soit dans toute la matrice, soit dans une seule équation.

1-2-2-3 PERTE MAXIMALE de PRECISION

Pour le programme en virgule flottante simple précision, nous avons prévu la détection de la perte maximale de précision au cours de la résolution d'un système linéaire. Par perte de précision, nous entendons le nombre de chiffres significatifs perdus lors de la soustraction de deux nombres voisins. Ainsi dans l'exemple suivant :

$$0,98243776 - 0,98242932 = 0,00000844$$

la perte est de 5 chiffres.

En fin de calcul, le contenu d'une mémoire nous indique la perte maximum de précision et l'élimination au cours de laquelle elle s'est produite. Ces renseignements éveillent l'attention et permettent de détecter des résultats dénués de toute signification. Des pertes de 1 ou 2 chiffres sont courantes et sans conséquence grave.

1-2-2-4 CALCUL en DOUBLE PRECISION

Enfin, le passage du programme de la méthode de GAUSS en virgule flottante simple précision à celui en double précision, ne présente aucune difficulté supplémentaire, la logique du programme restant inchangée. Il suffira seulement de prévoir un nombre deux fois plus grand de mémoires pour les données et les résultats intermédiaires ou définitifs ; il faudra manipuler des nombres occupant deux mémoires au lieu d'une seule et utiliser l'arithmétique virgule flottante en double précision.

1-2-3 NOMBRE d'OPERATIONS

Nous désignerons par « n » l'ordre du système linéaire à résoudre, et par « p » le nombre de séries de seconds membres pour lesquelles nous désirons obtenir la solution.

Supposons que nous nous apprêtions à éliminer la k° inconnue. L'équation pivot est composée de $(n - k + 1 + p)$ termes et sa division par le pivot nécessite :

$$(n - k + p) \text{ divisions.}$$

Ensuite, les calculs d'élimination concernent $(n - k)$ équations composées chacune de $(n-k + p+1)$ termes, dont le coefficient de l'inconnue à éliminer. Le calcul de chaque nouveau terme nécessite deux opérations (une multiplication et une addition). Pour le calcul des $(n - k + p)$ termes de chaque nouvelle équation, il faudra donc :

$$2 (n - k + p) \text{ opérations.}$$

et pour les $(n - k)$ nouvelles équations :

$$2 (n - k + p) (n - k)$$

Par suite, la k° élimination nécessite :

$$(n - k + p) (2 n - 2 k + 1)$$

L'obtention du système triangulaire sera effective après n éliminations de ce genre ; d'où le nombre correspondant d'opérations :

$$N_E = \sum_{k=1}^{n} (n - k + p)(2n - 2k + 1)$$

$$N_E = n \left[n (n+p) - \frac{n+1}{3} \times \frac{2n + 1}{2} \right]$$

La résolution du système triangulaire se fait par substitution inverse. Les calculs sont indépendants pour chaque série de seconds membres.

Pour une série de seconds membres, le calcul de la k° racine nécessite la connaissance des $(k - 1)$ premières racines. Le nombre d'opérations à effectuer est égal à : $2(k - 1)$. Le calcul des « n » racines nécessitera donc :

$$2 \sum_{k=1}^{n} (k - 1) = n (n - 1)$$

Finalement, pour « p » séries de seconds membres, la substitution inverse demande :

$$N_S = p\, n\, (n - 1) \text{ opérations}$$

Par conséquent, la résolution d'un système linéaire par la méthode de GAUSS nécessite au total

$$N = N_E + N_S = n \left[n (n + p) - \frac{n+1}{3} \times \frac{2n + 1}{2} \right] + p\, n\, (n - 1) \text{ opérations}$$

soit :

$$N = n\left[\frac{2n^2}{3} + \left(2p - \frac{1}{2}\right)n - \left(p + \frac{1}{6}\right)\right]$$

1-2-4 RESULTATS GENERAUX

1-2-4-1 PROGRAMMES en VIRGULE FLOTTANTE SIMPLE PRECISION

Les tableaux suivants nous donnent les limites d'utilisation, les temps de calcul et le nombre d'opérations pour la méthode de GAUSS programmée avec choix du pivot maximal dans toute la matrice ou sans choix de pivot.

Méthodes programmées en virgule flottante simple précision	Limites d'utilisation
GAUSS sans choix de pivot	$n = 39$ $p = 2$
GAUSS avec choix du pivot maximal dans toute la matrice	$n = 36$ $p = 2$

n	*Temps de calcul avec* p = 1		Nombre d'opérations (communaux deux méthodes).
	Méthode GAUSS avec choix du pivot dans toute la matrice. (simple précision)	Méthode GAUSS sans choix de pivot (simple précision)	
6	27 secondes	19 secondes	191
12	2 mm 38 s	2 minutes	1354
18	7 mn 56 s	6 mn 15 s	4353
24	17 mn 40 s	14 mn 20 s	10052
30	33 mn 09 s	27 mn 35 s	19315
36	56 mn 50 s	46 mn 58 s	33006

On constate évidemment que le programme sans choix de pivot est le plus rapide, Pour un nombre identique d'opérations arithmétiques, mais avec une logique simplifiée. En ce qui concerne la précision des résultats, pour un système linéaire à diagonale principale prédominante, le programme ne faisant aucun choix de pivot, elle est fonction de l'ordre dans lequel on présente les données. Par exemple, pour un système d'ordre 18 [Annexe II, 3] , résolu en simple précision, l'erreur maximale en valeur absolue peut varier de 10^{-4} à $4 . 10^{-7}$ pour des racines connues exactement et toutes inférieures en valeur absolue à 3,4. Par contre l'erreur maximale en valeur absolue reste toujours égale à $2 . 10^{-7}$ pour le même système linéaire, si le programme fait le choix du pivot maximal dans toute la matrice. Il semble donc que dans ce cas, cette dernière méthode soit préférable. Si l'on recommence les mêmes expériences avec un système linéaire mal conditionné, les résultats obtenus avec les deux méthodes sont aussi imprécis. Il n'est pas possible depréconiser une méthode plutôt que l'autre, seules des méthodes calculant en double précision ou du type « JORDAN-Itérations » sont alors à conseiller.

1-2-4-2 PROGRAMME en VIRGULE FLOTTANTE DOUBLE PRECISION

Les tableaux suivants donnent les limites d'utilisation, les temps de calculs et les nombres d'opérations arithmétiques pour la méthode de GAUSS avec choix du pivot maximal dans toute la matrice, programmée en virgule flottante double précision.

Méthode programmée en virgule flottante double précision	Limites d'utilisation
GAUSS avec choix du pivot maximal dans toute la matrice	$n = 24$ $p = 1$

n	Temps de calcul avec $p = 1$ Méthode de GAUSS avec choix du pivot maximal dans toute la matrice. (double précision)	Nombre d'opérations arithmétiques
6	1 s 11 s	791
12	8 s 09 s	1354
18	2[illegible] 02 s	4353
24		10052

L'utilisation de la méthode de GAUSS, avec choix du pivot maximal dans toute la matrice, programmée en virgule flottante double précision ne se justifie que pour la résolution des systèmes linéaires mal conditionnés. En effet, le temps de calcul est environ trois fois plus important que celui noté pour le même programme en simple précision et les possibilités maximales sont notablement réduites. [voir paragraphe 1-2-4-1].

A titre d'exemple, nous donnons sous formes de tableaux, les résultats obtenus pour un système mal conditionné d'ordre 6 [Annexe II, 5] à l'aide de la méthode de GAUSS avec choix du pivot maximal dans toute la matrice, programmée : soit en virgule flottante simple précision ; soit en virgule flottante double précision.

	R A C I N E S		
	x_1	x_2	x_3
valeurs exactes	8, 0000000	12, 000000	17, 000000
Résultats obtenus avec la méthode de GAUSS en Simple Précision	7, 9409000	12, 012494	17, 116815
Résultats obtenus avec la méthode de GAUSS en Double Précision	8, 0000000	11, 999999	16, 999999
	00013 40690	9999980163	9999715282

	R A C I N E S		
	x_4	x_5	x_6
valeurs exactes	- 15, 000000	- 0,30000000	- 30, 000000
Résultats obtenus avec la méthode de GAUSS en Simple Précision	- 15, 011952	- 0,30073770	- 30, 057937
Résultats obtenus avec la méthode de GAUSS en Double Précision	- 14, 999999	- 0, 29999999	- 29, 999999
	- 9999981549	- 9999869850	- 9999846540

Remarques

- L'unité en virgule flottante double précision s'écrit :

51 1000 0000 ⋮ 00 0000 0000

et nous l'écririons dans les tableaux précédents :

1, 0000000
0000000000

- Pour les deux méthodes de GAUSS utilisées, le choix du pivot maximal est fait dans toute la matrice.

- Nous trouverons les résultats obtenus pour l'exemple précédent à l'aide de la méthode de JORDAN, avec choix du pivot maximal dans toute la matrice, programmée : soit en virgule flottante simple précision, soit en virgule flottante double précision, au paragraphe 1-1-4-4.

1 - 3 AMELIORATION de la METHODE de JORDAN - METHODE de JORDAN-Itérations

1-3-1 PRINCIPE

Ayant obtenu par la méthode de JORDAN, une première solution approchée x_0 du système linéaire :

$$A x = b$$

nous déterminons le vecteur résidu

$$r_0 = A x_0 - b$$

$$= A x_0 - A x = A(x_0 - x)$$

et nous calculons la correction $\varepsilon_0 = x - x_0$ en résolvant le système linéaire $A \varepsilon_0 = - r_0$

La résolution de ce système est immédiate si l'on a pris soin de conserver en mémoire, tous les multiplicateurs utilisés au cours de chacune des « n » éliminations nécessaires pour la résolution de $A x = b$. Ceci conduit au stockage de $n (n - 1)$ nombres. D'autre part, la méthode ne peut être efficace que si nous calculons exactement les résidus, à partir des données et de la dernière approximation calculée, exprimée en virgule flottante simple précision.

Ayant obtenu ε_0, nous calculons une nouvelle approximation $x_1 = x_0 + \varepsilon_0$. Nous pouvons alors recommencer les mêmes calculs jusqu'à obtenir un vecteur correction ε_1 négligeable vis à vis de la précision utilisée pour les calculs.

1-3-2 PROGRAMMATION

Soit à résoudre un système linéaire d'ordre n avec p séries de seconds membres. Les données sont initialement enregistrées en mémoires :

- une première fois à partir d'une adresse M. Cet enregistrement sera détruit lors de la première résolution par la méthode de JORDAN.

- une seconde fois à partir de l' adresse $M + n^2 + p\,n$. Ce second enregistrement sera préservé

jusqu'à la fin des calculs et utilisé pour la détermination des résidus.

Il faut distinguer deux stades principaux dans la résolution, auxquels correspondent pour le programme :

- la première résolution (méthode de JORDAN, sans choix de pivot) avec mise en mémoire de tous les multiplicateurs.

- les itérations.

1-3-2-1 - PREMIERE RESOLUTION

Au cours de la première résolution, nous stockerons les multiplicateurs de chaque élimination à la place des coefficients relatifs à l'inconnue éliminée. Le pivot est pris sur la diagonale principale, ce qui ne nécessite aucun réarrangement des racines et des multiplicateurs en vue des calculs futurs. Cette façon de procéder conduit à un temps de calcul réduit, au détriment de la précision des résultats. Ceci n'est pas à considérer comme un inconvénient, la première résolution étant suivie d'itérations qui corrigeront les erreurs éventuelles. Les racines sont conservées dans «p n» mémoires que l'on place, par exemple, à la suite du second enregistrement des données (à partir de l'adresse $M + 2\ n\ (n + p)$). Ce premier travail nécessite $\frac{n}{2}(2n - 1)(n + 2p - 1)$ opérations arithmétiques élémentaires (Cf. méthode de JORDAN, paragraphe *1-1-3*).

1-3-2-2 ITERATIONS

La partie itérative débutera par le calcul des résidus ($2\ p\ n^2$ opérations) qui seront substitués aux seconds membres utilisés pour la première résolution. La résolution du nouveau système linéaire ($A\ \varepsilon_o = -r_o$) ne portera que sur les seconds membres. Elle sera donc très facilitée et ne nécessitera que $p\ n(2n - 1)$ opérations élémentaires. Cette résolution terminée, nous effectuons les corrections convenables sur l'approximation précédente ($p\ n$ opérations). Nous nous retrouvons alors dans les mêmes conditions qu'au début de cette seconde partie ; nous sommes prêts à commencer une nouvelle itération.

Remarques

Nous trouverons en annexe [Annexe I, 4] une étude détaillée du programme correspondant.

1-3-2-3 FORME du PROGRAMME

Ce programme a été essayé en virgule flottante simple précision. Il se présente sous forme de sous-programme, ce qui suppose le double enregistrement des données préalablement effectué. Son encombrement est de 400 mémoires auxquelles il faut ajouter les 200 mémoires réservées à l'arithmétique en Virgule Flottante simple précision. Il reste donc 1400 mémoires pour les $(2\ n + 3\ p)\ n$ mémoires nécessaires à l'enregistrement des données, au stockage des résultats intermédiaires et définitifs. De cette façon, avec $p = 1$, nous pourrons traiter avec ce sous-programme un système linéaire d'ordre maximum égal à 25.

Remarques

1 - Tous les calculs sont effectués en virgule flottante simple précision sauf celui des résidus. Ceux-ci sont calculés exactement, en double précision, à partir des données et des résultats exprimés en simple précision. A cet effet, un sous-programme spécial d'addition est inclus dans le sous-programme de résolution de systèmes linéaires.

2 - Lorsque les résultats de deux itérations successives sont absolument identiques, la dernière approximation calculée est considérée comme étant la solution recherchée. C'est-à-dire que l'on arrête les itérations lorsque toutes les corrections ε_i n'affectent pas les racines exprimées en virgule simple précision. La différence entre l'exposant de chaque racine et celui de la correction correspondante est alors au moins égale à 8.

1-3-3 NOMBRES d'OPERATIONS ARITHMETIQUES

Les resultats anoncés au parabraphe précédent nous permettent de dresser le tableau ci-dessous :

	Nombre d'opérations
Première résolution	$\frac{n}{2}(2n-1)(n+2p-1)$
Calcul des résidus	$2p\ n^2$
Calcul de la correction ε_i	$p\,n\,(2n-1)$
Calcul de la nouvelle approximation x_{i+1}	$p\,n$

Si la résolution d'un système linéaire nécessite K itérations, le nombre d'opérations arithmétiques correspondant sera :

$$N = \frac{n}{2}(2n-1)(n+2p-1) + (2p\,n^2 + p\,n\,(2n-1) + p\,n)(K+1)$$

En effet chaque itération, faisant suite à la première résolution, nécessite le calcul :

- des résidus,
- de la correction ε_i,
- de la nouvelle approximation x_{i+1}

et d'autre part, l'arrêt des calculs est décidé lorsque la dernière correction calculée ne modifie pas la dernière approximation obtenue.

Nous écrirons N sous une forme plus simple :

$$N = n\left[n^2 - \frac{3n}{2} + \frac{1}{2} + 4pnK + p(6n-1)\right]$$

Remarque

On pourrait étudier de la même façon que précédemment, l'amélioration de la méthode de GAUSS qui conduirait à la méthode de GAUSS-Itérations. Le sous-programme correspondant serait comparable à celui de la méthode de JORDAN-Itérations. Sa conception serait identique mais son encombrement serait plus important à cause de la programmation de la substitution inverse. Par contre, le nombre d'opérations arithmétiques néces-

saires à la détermination de la solution :

$$N' = n \left[\frac{2n^2}{3} - \frac{n}{2} - \frac{1}{6} + 4pnK + p(6n - 1) \right]$$

est inférieur au nombre N relatif à la méthode de JORDAN-Itérations. Le calcul du nombre d'opérations économisées (N - N') montre que l'économie réalisée est indépendante de p et K. Elle est égale à :

$\frac{n}{3}(n - 1)(n - 2)$, quantité définie et calculée dans le chapitre 3, paragraphe *3 - 1* (Remarques Générales).

Enfin, la précision des résultats fournis par les méthodes de GAUSS et de JORDAN étant comparables (voir paragraphe *3 - 1*) on ne peut pas espérer obtenir avec la méthode de GAUSS Itérations des résultats plus précis qu'avec celle de JORDAN Itérations.

1-3-4 RESULTATS

1-3-4-1 SITUATION de la METHODE de JORDAN Itérations par rapport aux Méthodes Directes Classiques

En dehors de toute considération sur la précision des résultats qui fera l'objet du paragraphe suivant (1-3-4-2), la méthode de JORDAN Itérations peut être comparée aux méthodes de JORDAN et de GAUSS programmées soit en virgule flottante simple précision, soit en virgule flottante double précision.

1-3-4-1-1 Comparaison avec les méthodes de JORDAN et de GAUSS programmées en virgule flottante simple précision.

Les trois méthodes comparées sont aussi simples à utiliser les unes que les autres comme on peut s'en rendre compte en consultant leurs notices d'utilisation [Annexe I, 1,3 et 4]. Par contre, dans tous les autres domaines (possibilités maximales, temps de calcul) la méthode de JORDAN Itérations s'avère inférieure à celles de JORDAN et de GAUSS, comme le montre le tableau suivant.

	Méthode de JORDAN Itérations	Méthode de JORDAN	Méthode de GAUSS
Possibilités maximales	$n = 25$ $p = 1$	$n = 37$ $p = 1$	$n = 36$ $p = 2$
Temps de Calcul pour $n = 6$; $p = 1$	51 secondes avec une seule itération	32 secondes	27 secondes
Nombre d'Opérations pour $n = 6$; $p = 1$	519 avec une seule itération	231	191

La méthode de JORDAN Itérations n'est donc pas à conseiller pour la résolution des systèmes linéaires «bien conditionnés».

1-3-4-1-2 Comparaison avec les méthodes de JORDAN et de GAUSS programmées en virgule flottante double précision.

Cette fois la méthode JORDAN Itérations supporte la comparaison.

1-3-4-1-2-1 **Comparaison** des encombrements et des possibilités maximales.

Le sous-programme de résolution de systèmes linéaires avec la méthode de JORDAN Itérations occupe au total 600 mémoires contre 645 avec la méthode de JORDAN en double précision et 680 avec celle de GAUSS en double précision.

Le stockage des données, des résultats intermédiaires et définitifs nécessite ($2n^2 + 3\ p\ n$) mémoires pour la première méthode et ($2\ n^2 + 2\ p\ n + 3\ n + 2\ p$) mémoires pour les deux autres.

Les deux encombrements précédents conduisent, pour chacun des trois programmes, à des possibilités maximales indiquées dans le tableau ci-dessous :

	Méthode de JORDAN Itérations	Méthode de JORDAN	Méthode de GAUSS
Possibilités Maximales	$n = 25$ $p = 1$	$n = 24$ $p = 2$	$n = 24$ $p = 1$

1-3-4-1-2-2 Comparaison des temps de calcul

La résolution d'un système linéaire d'ordre 6 avec une série de seconds membres, demande 1 minute 22 secondes par la méthode de JORDAN en Double précision et 1 minute 14 secondes par celle de GAUSS en double Précision.

Pour la méthode de JORDAN Itérations, le temps de calcul est variable avec le nombre d'itérations nécessaires :

avec aucune itération : 35 secondes

une itération : 51 secondes

deux itérations : 1 minute 07 secondes

trois itérations : 1 minute 23 secondes.

Les comparaisons que nous venons d'effectuer mettent en évidence les points communs aux méthodes de JORDAN et de GAUSS en Double Précision et celle de JORDAN Itérations. Il semble donc qu'on pourra les utiliser indifféremment pour résoudre les systèmes linéaires «mal conditionnés», sous réserve que les précisions des résultats obtenus soient comparables.

Remarques

La méthode de JORDAN Itérations présente l'avantage de mieux s'adapter au conditionnement du système linéaire traité, que les méthodes de JORDAN et de GAUSS en Double Précision. En effet, avec ces deux dernières méthodes, il faut prévoir à l'avance l'utilité ou l'inutilité d'effectuer les calculs en double précision.

Si l'on envisage le calcul d'une matrice inverse, la méthode la plus intéressante estcelle de JORDAN en simple ou en double précision, car elle ne nécessite aucune mémoire de travail supplémentaire. En effet, les colonnes de la matrice inverse sont progressivement introduites à la place laissée libre par les coefficients des inconnues éliminées, place que l'on utilise pour le stockage des multiplicateurs dans la méthode de JORDAN Itérations. La méthode de JORDAN est encore la plus rapide car elle nécessite le nombre minimum d'opérations arithmétiques (voir paragraphe *3-1*).

1-3-4-2 PRECISION des RESULTATS

Distinguons encore deux cas : les systèmes linéaires «bien conditionnés» etceux «mal conditionnés». Ceci revient en fait à comparer successivement la précision des résultats obtenus par la méthode de JORDAN Itérations avec celle des résultats fournis par les méthodes de JORDAN et de GAUSS programmées soit en simple précision, soit en double précision.

1-3-4-2-1 Cas des systèmes linéaires «bien conditionnés»

Les résultats obtenus par la méthode de JORDAN Itérations sont très satisfaisants. Avec des données exactes ou considérées comme telles, les résultats sont obtenus avec autant de décimales exactes que peut en fournir la simple précision. Nous avons donc systématiquement huit chiffres exacts pour tous les résultats.

Le gain de précision (au maximum les deux derniers chiffres significatifs) permis par la méthode de JORDAN Itérations est la contrepartie de possibilités maximales réduites et de temps de calculs plus importants, par rapport aux méthodes de JORDAN et de GAUSS programmées en virgule flottante simple précision.

Ce gain de précision ne peut pas justifier l'emploi systématique de la méthode de JORDAN Itérations pour la résolution des systèmes linéaires «bien conditionnés».

1-3-4-2-2 Cas des systèmes linéaires «mal conditionnés»

La précision des résultats obtenus peut varier, dans des limites importantes, d'un système linéaire à un autre. C'est ce que nous allons d'abord illustrer à l'aide de deux exemples ; puis en étudiant les causes de l'imprécision des résultats obtenus pour l'un d'eux, nous verrons s'il est possible d'améliorer la méthode de JORDAN Itérations.

1-3-4-2-2-1 Exemples de systèmes linéaires «mal conditionnés»

1-3-4-2-2-1-1 Exemple N° 1.- Il s'agit d'un système linéaire d'ordre 6 [Annexe II, 5] fourni par une méthode de moindres carrés. Nous trouverons dans les Remarques Générales (paragraphe 3-4-1) toutes les pertes de précision détectées au cours de la résolution de ce système à l'aide de la méthode de JORDAN programmée en virgule flottante simple précision.

Les résultats obtenus à l'aide de différentes méthodes sont résumés dans les tableaux suivants.

METHODES	RACINES		
	x_1	x_2	x_3
JORDAN Simple Précision Virgule Flottante	8, 4052073	11, 895020	16, 875200
JORDAN Double Précision Virgule Flottante (1)	8, 3 105260	11, 893953	17, 110095
JORDAN - ITERATIONS (2 Itérations)	8,3 105260	11,893953	17, 110095

METHODES	RACINES		
	$-x_4$	$-x_5$	$-x_6$
JORDAN Simple Précision Virgule Flottante	15, 533 90	0, 11500885	29, 492812
JORDAN Double Précision Virgule Flottante (1)	15, 534153	0, 11455551	29, 635235
JORDAN - ITERATIONS (2 itérations)	15, 534153	0, 11455551	29, 635235

(1) résultats arrondis à 8 chiffres significatifs.

1-3-4-2-2-1-2 Exemple N°2.- Nous considérons maintenant un système linéaire d'ordre 4 extrait d'une communication de Petear B. MADIC (Electrical Engineer, Institute «Boris Kidric», Belgrade, Yugoslavia») aux «Journées Internationales de Calcul Analogique» (Septembre 1955). Nous trouverons les données de ce système en annexe [Annexe II, 6] et les pertes de précision détectées lors de sa résolution à l'aide de la méthode de JORDAN programmée en virgule flottante simple précision dans les Remarques Générales au paragraphe *3-4-2.*

Les résultats obtenus à l'aide de différentes méthodes sont résumés dans les tableaux suivants.

METHODES	*RACINES*	
	x_1	x_2
Valeurs exactes	1, 0000000	1. 0000000
JORDAN Simple Précision Virgule Flottante	- 0, 48808465	1, 8970052
JORDAN Double Précision Virgule Flottante (1)	0. 99997655	1. 0000142
JORDAN - Itérations (2)	4. 1168511	- 0. 88249189

METHODES	*RACINES*	
	x_3	x_4
Valeurs exactes	1. 0000000	1. 0000000
JORDAN Simple Précision Virgule Flottante	1. 3793700	0. 77413045
JORDAN Double Précision Virgule Flottante (1)	1. 0000059	0. 99999651
JORDAN - Itérations (2)	0. 21724918	1. 4642657

(1) résultats arrondis à 8 chiffres significatifs

(2) Le nombre d'itérations effectuées est égal à 35. Les corrections ε_i ne deviennent pas négligeables, la méthode ne converge pas.

1-3-4-2-2-2 Amélioration de la méthode de JORDAN-Itérations

Les résultats fournis par la méthode de JORDAN-Itérations pour l'exemple N-1 sont excellents. Nous pouvons conclure de la même façon que pour les systèmes linéaires «bien conditionnés» (voir para-

graphe : *1-3-4-2-1*) : les résultats sont obtenus avec autant de décimales exactes que peut en fournir la virgule flottante en simple précision. Nous avons donc systématiquement 8 chiffres exacts pour tous les résultats.

Pour l'exemple N° II, seule la double précision donne de bons résultats, alors que la méthode de JORDAN Itérations me converge pas et fournit, après 35 itérations, des résultats sans signification. On peut tenter d'expliquer cette anomalie en considérant tous les multiplicateurs conservés en mémoire au cours de la première résolution du système linéaire. Et l'on constate qu'ils ont presque tous subi des pertes de précision (voir paragraphe *3-4-2*). Par conséquent, tous les calculs des itérations qui suivent la première résolution se font avec des valeurs erronées. C'est pourquoi nous avons pensé améliorer les résultats fournis par la méthode de JORDAN-Itérations en modifiant le programme initial [Annexe I, 4] de telle sorte que toute opération conduisant à une perte de précision soit réeffectuée immédiatement en virgule flottante double précision. Ceci revient en fait à calculer en double précision sur des données exprimées en simple précision ; les résultats obtenus étant conservés en simple précision. Ce mode opératoire est déjà utilisé dans le programme initial [Annexe I, 4] pour le calcul exact des résidus (r = Ax - b) à partir de données (A et b) et de résultats (x) exprimés en simple précision. Si bien que l'amélioration proposée pour le programme initial ne nécessite que 50 mémoires supplémentaires. Par contre, les temps de calcul seront augmentés dans des proportions variant d'un système linéaire à un autre ; les augmentations étant fonctions du nombre de pertes de précision rencontrées.

En utilisant le programme initial amélioré de la façon que nous venons de voir, la méthode de JORDAN Itérations converge pour l'exemple N° II et les résultats obtenus sont indiqués dans les tableaux ci-dessous.

Méthode de JORDAN-Itérations améliorée		
Valeurs exactes	$x_1 = 1,0000000$	$x_2 = 1,0000000$
Résultats obtenus après la première résolution	$x_1 =$ 1,0034382	$x_2 =$ 0,99791774
Résultats définitifs obtenus après deux itérations	$x_1 =$ 1,0036333	$x_2 =$ 0,99780559

Méthode de JORDAN-Itérations améliorée		
Valeurs exactes	$x_3 = 1,0000000$	$x_4 = 1,0000000$
Résultats obtenus après la première résolution	$x_3 =$ 0,99915395	$x_4 =$ 1,0004992
Résultats définitifs obtenus après deux itérations	$x_3 =$ 0,99908755	$x_4 =$ 1,0005412

Les résultats ainsi obtenus sont nettement améliorés mais ils sont moins précis que ceux fournis par la méthode de JORDAN programmée en virgule flottante double précision.

Nous voyons apparaître une autre anomalie : les résultats obtenus après deux itérations sont moins précis que ceux fournis par la première résolution. Ceci ne tient pas à la méthode de JORDAN-Itérations, mais à la structure du système linéaire. Pour en avoir une idée, considérons l'expression de l'erreur ε sur les racines en fonction des résidus r :

$$\varepsilon = - A^{-1} r$$

Nous avons calculé la matrice inverse A^{-1} à l'aide de la méthode de JORDAN programmée en virgule flottante double précision.

$$A^{-1} = \begin{bmatrix} 64979126 & -39245595 & -16318541 & 9678865 \\ -39245595 & 23703254 & 9855947 & -5845766 \\ -16318541 & 9855947 & 4098160 & -2430704 \\ 9678865 & -5845766 & -2430704 & 14417001 \end{bmatrix}$$

Les coefficients de cette matrice sont certainement entachés d'erreurs, mais nous sommes néanmoins sûrs de leurs ordres de grandeur et de leurs signes. A la vue de ces derniers résultats, l'on conçoit très bien que de faibles variations des résidus peuvent avoir sur l' erreur ε des influences absolument quelconques. Ainsi, si nous considérons les résidus après la première résolution et après les 2 itérations :

Résidus : Ax - b	
Après la première résolution	Après deux itérations
- 0, 04301642	0, 00157083
- 0, 05979997	0, 00218343
- 0, 06240685	0, 00228060
- 0, 05890203	0, 00215262

nous constatons qu'ils décroissent en valeur absolue et changent tous de signes alors que l'erreur ε augmente en valeur absolue. La méthode de JORDAN-Itérations ne peut rien contre cela.

Finalement, suivant le système linéaire considéré, la méthode de JORDAN-Itérations converge et donne d'excellents résultats, ou au contraire diverge. Dans ce dernier cas, les méthodes de JORDAN et de GAUSS programmées en virgule flottante double précision donnent les meilleurs résultats. Leur utilisation sera donc généralement préférable à celle de JORDAN-Itérations.

L'amélioration de la méthode de JORDAN-Itérations ne présente pas d'intérêt pratique ; mais elle montre qu'il est possible d'améliorer de la même façon les résultats fournis par les méthodes de JORDAN et de GAUSS programmées en virgule flottante simple précision, lors de la résolution de systèmes linéaires «mal conditionnés».

1 - 4 METHODE des GRADIENTS CONJUGUES

La méthode des Gradients Conjugués est dûe à HESTENES et STIEFEL. Elle est à la fois une méthode directe et indirecte. Elle est donc très séduisante à priori puisque, après n itérations (si n est l'ordre du système linéaire à résoudre) nous obtenons la solution recherchée, ce qui n'empêche nullement d'effectuer quelques itérations supplémentaires afin d'améliorer les résultats, pour compenser les erreurs d'arrondi.

1 - 4 - 1 PRINCIPE de la METHODE des GRADIENTS CONJUGUES

Le principe de la méthode des Gradients Conjugués est basé sur le fait que :

- la résolution du système linéaire $A x = b$ (A étant une matrice symétrique et définie positive).
- et la recherche de la valeur de x pour laquelle la forme quadratique $F(x) = \frac{1}{2} x' A x - b' x$ (ou x' et b' sont les transposées de x et b) est minimale, sont deux problèmes identiques.

Tous les points de l'espace à n dimensions pour lesquels la forme quadratique $F(x)$, relative à une matrice A : d'ordre n, symétrique et définie positive , est constante : constituent une surface du second degré d'équation

$$F(x) = x' A x - b' x = C$$

L'intérieur de ces surfaces comprend un point unique où la forme quadratique $F(x)$ prend sa valeur minimale. Les coordonnées de ce point constituent la solution à déterminer du système linéaire $A x = b$, et le centre des surfaces du second degré $F(x) = C$. Sa recherche se fera d'une façon itérative en passant successivement d'une surface de niveau C à des surfaces de niveaux inférieurs. Dans ce but, nous partons d'un point quelconque x_1 de l'espace à n dimensions, situé sur la surface du second degré $F(x) = C_1$. Par ce point x_1, nous faisons passer une direction quelconque, et nous cherchons la surface $F(x) = C_2$ ($C_2 < C_1$) tangente à cette direction. Ayant déterminé le point de contact x_2, nous recommençons avec celui-ci les mêmes opérations qu'avec le point x_1, et ainsi de suite .... On démontre qu'en opérant de cette façon, on converge vers le point cherché.

Il existe un certain nombre de méthodes basées sur ce principe (méthode de Relaxation, méthode de la plus grande pente) qui ne diffèrent les unes des autres que par le choix de la direction passant par les points successifs : $x_1, x_2, x_3, \ldots$.

Dans lecas de la méthode des GRADIENTS CONJUGUES, nous calculons le vecteur résidu r_1 au point de départ x_1. Nous faisons passer par ce dernier point, un plan défini par r_1 et une direction arbitraire d_1. Ce plan détermine un sous-espace à deux dimensions qui coupe les surfaces du second degré $F(x) = C$, suivant une série d'ellipses qui admettent le même centre. Nous prenons alors ce centre comme second point x_2 et nous recommençons avec lui les mêmes opérations qu'avec x_1 mais en prenant cette fois-ci une direction d_2 parfaitement déterminée : direction conjuguée de celle joignant les points x_1 et x_2, par rapport à l'ellipse pas-

sant par x_1 et de centre x_2. La direction d_2 est donc parallèle à la tangente en x_1 à l'ellipse passant par ce point.

On démontre qu'en opérant de cette façon, le $n^{ème}$ point calculé : x_n, est le point recherché, c'est-à-dire le centre des surfaces du second degré, F (x) = C. En réalité, on n'obtient pas exactement ce point à cause des erreurs d'arrondi.

Tout ce que nous venons de voir, se traduit du point de vue numérique, par la chaîne de calculs suivante, dans le cas d'une matrice A symétrique et définie positive :

$$(1) \qquad r_1 = b - A x_1$$

$$(2) \qquad p_1 = r_1$$

$$(3) \qquad a_1 = \frac{\left| r_1 \right|^2}{p'_1 A p_1}$$

$$(4) \qquad x_2 = x_1 + a_1 p_1$$

$$(5) \qquad r_2 = r_1 - a_1 A p_1$$

$$(6) \qquad b_1 = \frac{\left| r_2 \right|^2}{\left| r_1 \right|^2}$$

$$(7) \qquad p_2 = r_2 + b_1 p_1$$

Nous poursuivons ensuite les calculs en reprenant la chaîne ci-dessus à la relation (3), en augmentant tous les indices d'une unité, et ainsi de suite ..... jusqu'à obtenir x_n. La valeur de départ x_1 sera prise nulle à moins que nous connaissions une approximation de la solution recherchée.

Remarque

La notation p'_i représente la transposée de la matrice colonne p_i.

D'autre part, $|r_i|^2$ représente la somme des carrés de toutes les composantes du vecteur colonne r_i.

GENERALISATION de la METHODE des GRADIENTS CONJUGUES

Il est possible de généraliser la méthode des GRADIENTS CONJUGUES, pour l'appliquer à la résolution de systèmes linéaires dont la matrice A n'est pas symétrique et définie positive. Cette généralisation consiste à remplacer le système linéaire à résoudre :

$$A x = b$$

par un nouveau système :

$$A' A x = A' b$$

admettant la même solution et ayant une matrice AA' symétrique et définie positive. (A' représente la transposée de la matrice A). Nous pouvons alors opérer de deux façons différentes :

- appliquer au nouveau système, la chaine de calculs précédente. Cette première méthode présente l'inconvénient d'utiliser la matrice A' A qui sera entâchée d'erreurs d'arrondi.

- ou bien, utiliser la nouvelle forme prise par la chaîne de calculs précédente, lorsque nous l'appliquons au nouveau système à résoudre. HESTENES et STIEFEL ont déterminé cette nouvelle chaîne de calculs qui se présente sous la forme suivante :

$$(I) \qquad r_1 = b - A x_1$$

$$(II) \qquad p_1 = A' r_1$$

$$(III) \qquad a_1 = \frac{|A' r_1|^2}{|A p_1|^2}$$

$$(IV) \qquad x_2 = x_1 + a_1 p_1$$

$$(V) \qquad r_2 = r_1 - a_1 A p_1$$

$$(VI) \qquad b_1 = \frac{|A' r_2|^2}{|A' r_1|^2}$$

$$(VII) \qquad p_2 = A' r_2 + b_1 p_1$$

Nous poursuivons alors les calculs en reprenant la chaîne ci-dessus à la relation (III), en augmentant tous les indices d'une unité ; et ainsi de suite .... jusqu'à obtenir x_n. La valeur de départ x_1 sera choisie de la même manière que pour la chaîne originale.

Pour une étude approfondie de la méthode des GRADIENTS CONJUGUES et des méthodes basées sur le même principe, nous renvoyons à la bibliographie sommaire placée en annexe (Annexe III, 5 et 8)

1 - 4 - 2 PROGRAMMATION

Nous nous sommes bornés à étudier la méthode des GRADIENTS CONJUGUES appliquée à la résolution des systèmes linéaires à matrice quelconque. Pour cela nous avons initialement utilisé la deuxième chaîne de calculs indiquée ci-dessus. Mais ayant constaté des écarts importants entre la valeur réelle des résidus et celle calculée à partir de la relation (V), nous avons substitué à cette dernière le calcul direct des résidus à partir de leur définition

$$(V') \qquad r_i = b - A x_i$$

Nous prenons comme valeur de départ un vecteur x_1 nul ou une approximation connue de la solution recherchée, D'autre part, il faut noter qu'il est possible de reprendre les calculs à leur début, à n'importe quel moment de la résolution. Nous prenons alors la dernière approximation calculée comme vecteur de départ x_1.

La programmation de la méthode des GRADIENTS CONJUGUES ne pose pas de difficulté. Elle est certainement beaucoup plus aisée que la programmation des méthodes de GAUSS et JORDAN avec choix du pivot dans toute la matrice. Le programme de la méthode des GRADIENTS CONJUGUES se compose essentielle-

ment de quatre sous-programmes et d'un programme de liaison qui organise l'enchaînement des divers calculs. Nous aurons :

- un premier sous-programme chargé d'effectuer le calcul

$$M_1 - M_2 M_3$$

où M_1, M_2 et M_3 représentent des matrices. Nous utiliserons ce sous-programme pour le calcul

- des résidus (formule V')
- des vecteurs $A p_i$ et $A' r_i$

- un second sous-programme calculant :

$$V_1 + q V_2$$

où V_1, V_2 représentent des vecteurs et q un scalaire. Nous utiliserons ce sous-programme pour le calcul des vecteurs p_i (VII) et x_i (formule IV).

- un troisième sous-programme calcule les coefficients a_i (formule III) et b_j (formule VI).

- finalement, un sous-programme de perforation que nous utiliserons pour prendre connaissance des résultats définitifs et éventuellement du résultat de chacune des itérations.

1-4-3 NOMBRE d'OPERATIONS ARITHMETIQUES.

La chaîne de calculs utilisée étant celle qui vient d'être indiquée, nous pouvons calculer le nombre d'opérations arithmétiques qu'il faut effectuer pour résoudre un système linéaire d'ordre n, donné avec p séries de seconds membres.

Examinons tout d'abord, le nombre d'opérations nécessaires au calcul des éléments de chaque itération, dans le cas général et pour un seul vecteur second membre.

1-4-3-1 VECTEUR RESIDU

Nous avons le vecteur résidu $\vec{r}$ défini par :

$$r = b - Ax$$

Le calcul de chacune des n composantes :

$$r_j = b_j - \sum_{i=1}^{n} a_{ji} x_i$$

nécessite : 2 n opérations (n multiplications et n additions). Le calcul des « n » composantes sera effectif après :

$$2\ n^2 \text{ opérations.}$$

1-4-3-2 VECTEUR Ar

Il s'agit d'un calcul analogue à celui du vecteur résidu en y faisant $b = \vec{O}$. Il faudra donc encore : « $2\ n^2$ » opérations.

1-4-3-3 CALCUL de $|Ar|^2$

Il faut calculla somme des carrés des « n » composantes du vecteur Ar, ce qui nécessite 2 n opé-

rations. En effet, lors du cumul des carrés des composantes, il faudra additionner le premier à zéro.

1-4-3-4 CALCUL de b

Nous avons $b_{j-1} = \frac{|Ar_j|^2}{|Ar_{j-1}|^2}$. Une seule oprération sera donc suffisante.

1-4-3-5 CALCUL de p

Ce vecteur est défini par :

$$p_j = Ar_j + b_j p_{j-1}$$

Le calcul de ses «n» composantes nécessitera donc $2n$ opérations.

1-4-3-6 VECTEUR Ap

Le calcul de ce vecteur est comparable à celui du vecteur Ar . Il nécessitera donc «$2n^2$» opérations.

1-4-3-7 CALCUL de $|Ap|^2$

Calcul identique à celui de $|Ar|^2$, donc : «$2n$» opérations.

1-4-3-8 CALCUL de a

Nous avons : $aj = \frac{|Ar_j|^2}{|Ap_j|^2}$. Une seule opération sera donc suffisante.

1-4-3-9 CALCUL du VECTEUR SOLUTION : x

Il est défini par :

$$x_{j+1} = x_j + a_j p_j$$

Le calcul de ses «n» composantes nécessitera donc «$2n$» opérations.

Pour la 1° itération, le vecteur initial x_1 est nul, le calcul du vecteur résidu $\vec{r}$ ne nécessite aucune opération. Les calculs à effectuer sont ceux relatifs aux paragraphes : 1-4-3-2, 1-4-3-3 , 1-4-3-6 , 1-4-3-7 , 1-4-3-8 et 1-4-3-9. La première itération nécessite donc finalement :

$$(4n^2 + 6n + 1) \text{ opérations.}$$

Pour toutes les autres itérations, les calculs à effectuer sont ceux de tous les paragraphes : 1-4-3-1 à 1-4-3-9 , soit au total :

$$(6n^2 + 8n + 2) \text{ opérations .}$$

Par suite, si nous avons «p» vecteurs seconds membres, et si nous effectuons K itérations pour obtenir chacun des «p» vecteurs solutions ; il faudra effectuer :

$N = p\,[(4n^2 + 6n + 1) + (K - 1)(6n^2 + 8n + 2) + 2n^2)]$ opérations car le calcul de chaque vecteur solution, se termine par celui du vecteur résidu correspondant.

$$\boxed{N = p\,[2K(3n^2 + 4n + 1) - (2n + 1)]}$$

Remarque

Généralement, nous prendrons : $K = n + 1$. Dans ce cas, nous aurons :

$$\boxed{N = p(6n^3 + 14n^2 + 8n + 1)}$$

1-4-4 RESULTATS

1-4-4-1 RESULTATS GENERAUX

Dès les premiers essais du programme de la méthode des GRADIENTS CONJUGUES effectuant tous les calculs en virgule flottante, nous avons mis en évidence l'inconvénient majeur de cette méthode : l'importance des temps de calcul.

A titre d'exemple, donnons le tableau ci-dessous :

Temps de résolution d'un système linéaire d'ordre n , donné avec une seule série de seconds membres

n	*Méthodes utilisées*		
	Gradients conjugués	Jordan	Gauss
6	2 m. 43 s.	32 secondes	27 secondes
12	18 m. 11 s.	3 m. 28 s.	2 m. 38 s.

On constate que d'une façon générale, ces temps sont environ 5 fois supérieurs à ceux mesurés pour les méthodes directes de JORDAN et de GAUSS. Ceci se comprend aisément quand on considère le tableau suivant où nous faisons figurer les nombres d'opérations arithmétiques nécessaires à la résolution de systèmes linéaires à l'aide de ces différentes méthodes :

Nombre d'opérations nécessaires pour résoudre un système linéaire donné avec une seule série de seconds membres

Ordre du système à résoudre	*Méthodes utilisées*		
	Gradients Conjugués	Jordan	Gauss
6	1849	231	191
12	12 481	1794	1354
18	39 673	5985	4353
24	91 201	14 100	10 052
30	174 841	27 435	19 315
36	298 369	47 286	33 006

Nous constatons que la méthode des GRADIENTS CONJUGUES entraîne 6,3 à 8 fois plus d'opérations arithmétiques que la méthode de JORDAN, et de 9 à 9,7 fois plus que celle de GAUSS. L'écart constaté entre les rapports des temps et des opérations provient du fait que dans la méthode des GRADIENTS CONJUGUES : les opérations logiques sont très réduites alors que dans les méthodes de JORDAN et de GAUSS, il faut choisir le pivot et réordonner les résultats à chaque élimination. Cette même constatation explique que l'on rencontre peu de difficultés lors de la programmation de la méthode des GRADIENTS CONJUGUES.

1-4-4-2 PROGRAMMATION en VIRGULE FIXE

Nous avons pensé à améliorer le programme de la méthode des GRADIENTS CONJUGUES en effectuant tous les calculs en virgule fixe, avec un cadrage commun à toutes les données et tous les résultats intermédiaires. En effet, la majorité des calculs consiste en des produits de matrices ou de matricespar des vecteurs, et ce genre de calculs est le mieux adapté à la programmation en virgule fixe car il se ramène à des cumuls de produits ayant le même cadrage. Seul le calcul terminal des résidus qui fait intervenir un cumul de nombres ayant des cadrages différents, fait exception : il s'agit, en effet, de soustraire à chaque second membre b_i (i^e composante du vecteur b), le cumul convenable de la forme $\sum_{j=1}^{n} a_{ij}\ x_j$. Ce calcul doit être fait avec le maximum de précision, ce qui conduit à adopter pour les résidus un cadrage variant à chaque itération, les résidus devenant de plus en plus petits au fur et à mesure que les calculs avancent. L'on perd alors la plus grande partie des avantages de la virgule fixe car l'on est pratiquement ramené à faire les mêmes opérations qu'en virgule flottante. La programmation en virgule fixe est beaucoup plus difficile qu'en virgule flottante surtout si nous nous limitons pour les données et résultats intermédiaires, au même nombre de mémoires qu'en virgule flottante tout en recherchant à manipuler des coefficients ayant un nombre maximum de chiffres significatifs. Ceci peut conduire en cours de calcul à des recadrages de résultats intermédiaires, par exemple si tous les coefficients manipulés diminuent. C'est précisement le cas des résidus et éventuellement celui de résultats calculés à partir de ces résidus. Un problème analogue de recadrage est à considérer lorsqu'au contraire tous les coefficients manipulés augmentent. En opérant de cette façon en virgule fixe, nous avons constaté un gain de temps important, de l'ordre de 50%. Par exemple le temps de résolution d'un système linéaire d'ordre 6 avec une seule série de second membre et en effectuant 7 itérations, passe de 2 mn 43 s en virgule flottante à 1 mn 25 s en virgule fixe. Néanmoins, les temps de calculs restent touj ours nettement supérieurs à ceux enregistrés pour les méthodes de JORDAN et de GAUSS ; ce qui s'explique puisque, si nous pouvons toujours accélerer les calculs en améliorant la programmation, il est impossible de contrecarrer le fait que la méthode des GRADIENTS CONJUGUES nécessite de 6 à 9 fois plus d'opérations que les méthodes de JORDAN et de GAUSS.

Sur le plan des possibilités maxima, la méthode des GRADIENTS CONJUGUES est comparable aux autres méthodes classiques comme le montre le tableau ci-après :

METHODES	LIMITES d'UTILISATION
GRADIENTS CONJUGUES en virgule flottante	$n = 35$ $p = 1$
GRADIENTS CONJUGUES en virgule fixe	$n = 35$ $p = 2$
JORDAN en virgule flottante avec choix du pivot	$n = 37$ $p = 1$
GAUSS en virgule flottante avec choix du pivot	$n = 36$ $p = 2$

1-4-4-3 NOMBRE D'ITERATIONS - PRECISION des RESULTATS

Pour tous les systèmes linéaires que nous avons étudiés à l'aide de la méthode des GRADIENTS CONJUGUES programmée en virgule flottante ou en virgule fixe, nous avons perforé les résultats de toutes les itérations. Chaque fois que nous avons pris un vecteur nul comme vecteur de départ x_1, nous avons constaté qu'il ne fallait pas espérer obtenir la solution du système linéaire avant d'avoir effectué n itérations ; ce qui est parfaitement en accord avec la théorie de la méthode.

Nous avons encore constaté qu'en effectuant la $(n + 1)^\circ$ itération, les résultats étaient souvent améliorés. Ce fait est d'ailleurs signalé dans le mémoire d'HESTENES et STIEFEL (Annexe III, 8). Par contre, il est inutile de continuer les calculs au-delà de la $(n + 1)^\circ$ itération, aucune amélioration des résultats n'étant constatée. Nous pouvons expliquer en partie ce phénomène, par le fait que les résidus ne sont pas calculés exactement (ils sont calculés en simple précision à partir de données en simple précision) ; par suite, comme ils sont généralement très faibles à ce stade du calcul, ils seront toujours entâchés d'erreurs qui ont pris de plus en plus d'importance avec l'avancement des calculs. Il ne faut pas non plus négliger l'influence des erreurs dûes aux arrondis.

A titre d'exemple, voici pour un système linéaire d'ordre 6 [Annexe II, 1] dont les racines exactes en simple précision sont connues et sont toutes inférieures à 3,1 en valeur absolue : les erreurs absolues maximales relatives aux résultats des 10 premières itérations pour un vecteur de départ nul.

N° d'ordre de l'itération	Erreur maximale absolue	N° d'ordre de l'itération	Erreur maximale absolue
1	0,18	6	$3 . 10^{-8}$
2	0,07	7	$1 . 10^{-8}$
3	0,02	8	$1 . 10^{-8}$
4	0,003	9	$3 . 10^{-8}$
5	0,002	10	$6 . 10^{-8}$

Nous noterons la très grosse amélioration enregistrée lors du passage de la 5° à la 6° itération. A partir de cette dernière itération, les erreurs relevées portent sur la dernière décimale fournie par la virgule flottante en simple précision.

Vis à vis des autres méthodes classiques (JORDAN, GAUSS), les résultats sont obtenus avec des précisions comparables pour les systèmes linéaires «bien conditionnés». Pour les systèmes «mal conditionnés», la méthode des GRADIENTS CONJUGUES conduit, elle aussi, à des résultats sans signification, qu'il n'a pas été possible d'améliorer en augmentant le nombre d'itérations.

Il serait possible d'améliorer la méthode des GRADIENTS CONJUGUES en calculant exactement tous les résidus (calculs en double précision sur des données en simple précision). On pourrait alors espérer : réduire le nombre des itérations dans le cas des systèmes «bien conditionnés» , et rendre efficaces les itérations supplémentaires (ordre supérieur à $n + 1$) dans le cas des systèmes «mal conditionnés» . Sur le plan pratique, cette modification ne s'impose pas étant donné qu'elle augmenterait encore les temps de calcul qui sont déjà un gros handicap pour la méthode.

1-4-4-4 APPLICATION à la RESOLUTION des SYSTEMES LINEAIRES à MATRICE-BANDE

La méthode des GRADIENTS CONJUGUES possède une propriété qui peut être intéressante dans certains cas. Contrairement aux méthodes de JORDAN et de GAUSS qui procèdent par destruction des données la méthode des GRADIENTS CONJUGUES les conserve intactes d'un bout à l'autre des calculs de résolution. Ceci est très intéressant pour la résolution des systèmes linéaires à matrice-bande. Nous pourrons alors nous contenter de n'enregistrer en mémoires, que les coefficients de la bande, à priori non nuls ; ce qui permet d'envisager le traitement de systèmes d'ordre élevé. La logique du programme correspondant se complique par rapport au programme classique, du fait que les coefficients de la bande de la matrice sont enregistrés dans des mémoires consécutives, mais la chaîne de calculs reste inchangée.

Sans entrer dans les détails, nous allons nous contenter d'exposer les principales caractéristiques du programme de résolution des systèmes linéaires à matrice bande par la méthode des GRADIENTS CONJUGUES. Ce programme est directement issu du programme relatif aux matrices quelconques en ne modifiant que

le sous-programme qui travaille sur la matrice bande du système.. (voir paragraphe 1-4-2). Sa limite d'utilisation est définie par l'inégalité :

$$(m+p+4)\,n - \frac{m^2-1}{4} \leq 1349$$

si nous appelons :

n : l'ordre du système linéaire

p : le nombre de séries de seconds membres.

m : le nombre de diagonales composant la bande de la matrice. Cette quantité est obligatoirement impaire.

Ainsi par exemple, avec $p = 1$ et $m = 3$, nous calculons $n_{MAX} = 168$. Il est donc possible de traiter des systèmes d'ordre très élevé.

Les calculs ne portant que sur les coefficients, à priori non nuls, de la bande ; le nombre d'opérations arithmétiques nécessaires à l'obtention de la solution devient pour $(n + 1)$ itérations :

$$\left\{(n+1)\left[6mn + 8n - \frac{3(m^2-1)}{2}\right] + 1\right\} p$$

Il est nettement inférieur à celui calculé dans le cas d'une matrice quelconque ; les temps de calcul se trouveront donc réduits dans la même proportion , comme nous le constatons dans le tableau ci-dessous :

Caractéristiques du système linéaire			Méthode utilisée ($n + 1$ itérations)			
			Gradients Conjugués pour les systèmes linéaires à MATRICE BANDE		Gradients Conjugués pour les systèmes linéaires à MATRICE QUELCONQUE	
n	m	p	Temps de calcul	Nombre d'opérations	Temps de calcul	Nombre d'opérations
6	3	1	1 mn 40 s.	1009	2 mn 43 s.	1705
12	7	1	11 mn 10 s.	6865	18 mn 11 s.	11905

Les temps de calculs et les nombres d'opérations sont divisés par environ 1,7.

Les remarques sur la précision des résultats faites précédemment, à propos du programme relatif aux systèmes linéaires à matrice quelconque, restent valables pour le programme relatif aux systèmes linéaires à matrice bande. Toutefois dans ce cas particulier, nous avons généralement une matrice à diagonale principale prédominante, aussi il est intéressant de prendre comme vecteur de départ x_1, les seconds membres divisés par le terme correspondant de la diagonale principale de la matrice bande.

1-4-4-5 CONCLUSION

Nous pensons que la méthode des GRADIENTS CONJUGUES n'est pas à conseiller pour la résolution de tous les systèmes linéaires, dans le cas où nous sommes obligés de prendre un vecteur de départ x_1 quelcon que (par exemple nul) ; ceci à cause de ses temps de calcul beaucoup plus élevés que ceux des méthodes de JORDAN et de GAUSS (systèmes linéaires à matrice quelconque) ou celui de la méthode de JORDAN PROGRESSIVE (systèmes linéaires à matrice bande) et malgré la bonne précision des résultats obtenus après $(n + 1)$ itérations.

La méthode des GRADIENTS CONJUGUES devient intéressante, et peut être utilisée, chaque fois que l'on connait à priori, une approximation de la solution recherchée.

Dans le cas des systèmes linéaires «mal conditionnés» , la méthode des GRADIENTS CONJUGUES ne permet pas, comme toutes les autres méthodes entièrement programmées en simple précision, d'obtenir un résultat valable. Nous serons obligés d'avoir recours à une méthode directe. (JORDAN ou GAUSS) programmée en double précision ou à la méthode de JORDAN-Itérations. Il est en effet difficile d'envisager la programmation en double précision de la méthode des GRADIENTS CONJUGUES à cause des temps de calculs qui deviendraient prohibitifs.

Chapitre 2

METHODES PARTICULIERES

2 - 1 METHODE de JORDAN PROGRESSIVE

2-1-1 EXPOSE THEORIQUE

2-1-1-1 PRINCIPE de la METHODE

Le principe de la méthode de JORDAN Progressive est le même que celui de la méthode de JORDAN classique, seule la technique de l'élimination est différente. Il faudra encore éliminer successivement toutes les inconnues pour obtenir finalement une «matrice diagonale»

2-1-1-1-1 Technique de l'élimination

Les définitions et les propriétés du pivot et de l'équation pivot restant les mêmes que pour la méthode de JORDAN classique (voir paragraphes *1-1-1-2-1* et *1-1-1-2-2*), soit à résoudre le système linéaire d'ordre n :

$$a_{11} x_1 + a_{12} x_2 + a_{13} x_3 + \ldots + a_{1n} x_n = b_1$$

$$a_{21} x_1 + a_{22} x_2 + a_{23} x_3 + \ldots + a_{2n} x_n = b_2$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

$$a_{n1} x_1 + a_{n2} x_2 + a_{n3} x_3 + \ldots + a_{nn} x_n = b_n$$

Nous commençons les calculs d'élimination en considérant seulement la première équation. Elle servira d'équation pivot pour l'élimination de la première inconnue x_1, aussi nous la divisons immédiatement par le pivot (coefficient de x_1). Dans la méthode de JORDAN classique, on élimine alors la première inconnue dans les $(n-1)$ autres équations. Ici nous nous contentons de l'éliminer dans la seconde équation. A cet effet, nous soustrayons de cette équation, l'équation pivot multipliée par le coefficient de x_1 (multiplicateur pris dans la deuxième équation. La nouvelle équation ainsi obtenue servira d'équation pivot pour l'élimination de la 2° inconnue x_2, aussi nous la divisons immédiatement par le pivot (coefficient de x_2) et nous éliminons la 2° inconnue dans la première équation du moment. A cet effet, nous soustrayons de cette équation, l'équation pivot multipliée par le coefficient de x_2 (multiplicateur) pris dans la première équation du moment. Le système linéaire se présente alors sous la forme :

$$x_1 + o x_2 + a^2_{13} x_3 + a^2_{14} x_4 + \ldots\ldots + a^2_{1n} x_n = b^2_1$$

$$o x_1 + x_2 + a^2_{23} x_3 + a^2_{24} x_4 + \ldots\ldots + a^2_{2n} x_n = b^2_2$$

les $(n-2)$ autres équations restent inchangées.

Nous introduisons alors la troisième équation dans laquelle nous éliminons les deux premières inconnues à l'aide des deux premières équations comme équations pivots. Les multiplicateurs utilisés sont respectivement les coefficients de x_1 et x_2 pris dans la troisième équation. La nouvelle équation ainsi obtenue

est divisée par son coefficient de la troisième inconnue. Elle servira d'équation pivot pour éliminer cette inconnue, ce que nous faisons immédiatement dans les deux premières équations. Les multiplicateurs utilisés sont les coefficients de x_3 pris dans ces équations. Nous poursuivons alors les calculs de la même façon, en introduisant une à une les $(n-3)$ équations restantes.

Lorsque nous nous apprêtons à traiter la k° équation, le système a la forme suivante :

$$x_1 + ox_2 + ox_3 + ox_4 + \ldots\ldots + ox_{k-1} + a^{k-1}_{1,k}\, x_k + \ldots\ldots + a^{k-1}_{1,n}\, x_n = b^{k-1}_1$$

$$ox_1 + x_2 + ox_3 + ox_4 + \ldots\ldots + ox_{k-1} + a^{k-1}_{2,k}\, x_k + \ldots\ldots + a^{k-1}_{2,n}\, x_n = b^{k-1}_2$$

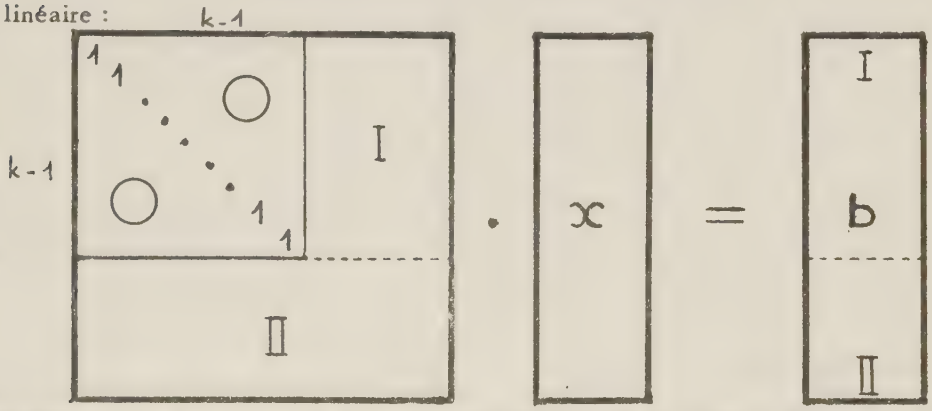
$$ox_1 + ox_2 + x_3 + ox_4 + \ldots\ldots + ox_{k-1} + a^{k-1}_{3,k}\, x_k + \ldots\ldots + a^{k-1}_{3,n}\, x_n = b^{k-1}_3$$

$$ox_1 + ox_2 + \ldots\ldots\ldots + ox_{k-2} + x_{k-1} + a^{k-1}_{k-1,k}\, x_k + \ldots\ldots + a^{k-1}_{k-1,n}\, x_n = b^{k-1}_{k-1}$$

$$a_{k_1} x_1 + a_{k_2} x_2 + \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots + a_{k,n}\, x_n = b_k$$

$$a_{n1} x_1 + a_{n2} x_2 + \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots + a_{n,n}\, x_n = b_n$$

Les $(n - k + 1)$ dernières équations sont inchangées. On peut représenter schématiquement le système linéaire :



Les zones référencées I contiennent de nouveaux coefficients. Les zones II contiennent encore les coefficients initiaux. La matrice unité d'ordre $(k-1)$ qui apparaît dans la partie supérieure gauche de la matrice du système, progresse horizontalement vers la droite et verticalement vers le bas.

Immédiatement après le traitement de la n° équation, la matrice du système sera une matrice unité d'ordre n. Les solutions recherchées sont disponibles, il s'agit des derniers seconds membres calculés.

2-1-1-1-2 Mode opératoire

Chaque étape du calcul peut être décomposée comme suit :

- 1 - prise en considération de la k° équation.

- 2 - élimination de ses $(k-1)$ premières inconnues. Les équations pivots utilisées sont successivement les $(k-1)$ premières équations. Le multiplicateur de chaque élimination est le coefficient de la racine x_i convenable $(1 \leqslant i \leqslant k-1)$, pris dans la k° équation. Etant donnée la forme des équations pivots, les calculs de chacune de ces éliminations ne modifient que les seconds membres et les termes de rang supérieur à $(k-1)$ du premier membre de la k° équation.

- 3 - division de la nouvelle équation ainsi obtenue par son coefficient de la k° inconnue. Cette opération n'intéresse que les termes de rang supérieur à k du premier membre et les seconds membres de la k° équation.

- 4 - élimination de la k° inconnue dans les (k - 1) premières équations. L'équation pivot est la k° équation prise sous sa dernière forme. Les multiplicateurs sont les coefficients de la k° inconnue pris successivement dans les (k - 1) premières équations.

Tous ces calculs ne modifient que les termes de rang supérieur à k dans le premier membre et les seconds membres de chacune des équations intéressées.

Il faut noter qu'au tout début du calcul, lorsque k = 1, seules les 1° et 3° opérations précédentes sont à effectuer. On fait immédiatement après k = 2, et toutes les opérations précédentes seront effectuées à chaque étape du calcul, jusqu'à faire k = n + 1.

2-1-1-1-3 Remarques

1/ La technique d'élimination de la méthode de JORDAN Progressive peut être caractérisée par le fait que les équations sont introduites une à une, et qu'à chaque étape on élimine le nombre le plus grand possible d'inconnues dans les équations prises en charge, à l'aide de ces mêmes équations.

Nous verrons au paragraphe «Programmation» comment cette propriété permet d'envisager le traitement des systèmes d'ordre élevé (n >36).

2/ D'après le mode opératoire indiqué ci-dessus, les inconnues sont éliminées dans l'ordre où elles se présentent. Le pivot est pris d'office sur la diagonale principale de la matrice du système.

Le choix du pivot maximal dans une seule équation est théoriquement possible. La technique de l'élimination reste inchangée. Mais une fois tous les calculs terminés, il faudra réarranger les résultats, pour mettre la matrice unité d'ordre n sous sa forme habituelle. Ce réarrangement se fera à l'aide de permutations d'équations seulement.

Le choix dupivot maximal dans toute la matrice nécessite la prise en charge initiale de toutes les équations. En utilisant toujours la même technique d'élimination, on prend encore les équations une à une, mais dans un ordre quelconque. Un réarrangement des résultats, par permutation d'équations, sera nécessaire en fin de calcul. Ce procédé n'est pas à conseiller comme nous l'expliquerons dans le paragraphe suivant (PROGRAMMATION : *2-1-3*).

2 - 1 - 2 NOMBRE d'OPERATIONS

Nous désignerons par «n» l'ordre du système linéaire à résoudre, et par «p» le nombre de séries de seconds membres pour lesquelles nous désirons obtenir la solution.

Supposons que nous abordions la k° élimination. Elle commence par l'élimination des (k - 1) premières inconnues de la k° équation. Au cours de l'élimination de chacune de ces inconnues, nous avons (n + p - k + 1) termes à calculer. Le calcul de chacun d'eux est de la forme «a + mb» ; il nécessite donc deux opérations. L'élimination des (k - 1) premières inconnues de la k° équation nécessitera finalement :

2 (k - 1) (n + p - k + 1) opérations .

La k° élimination se poursuit par la division de l'équation pivot (la k° équation débarassée de ses (k - 1) premières inconnues) par son terme diagonal. Cette opération se traduit par :

$$(n + p - k) \text{ divisions.}$$

Enfin nous terminons la k° élimination par l'élimination de la k° inconnue dans les (k - 1) premières équations. Pour chaque équation, Il faut calculer $(n + p - k)$ termes de la forme « a + mb » ; ce qui nécessitera :

$$2 (n + p - k) \text{ opérations ,}$$

Pour les (k - 1) équations, nous aurons :

$$2 (k - 1) (n + p - k) \text{ opérations.}$$

En définitive, la k° élimination nécessitera au total

$$N_k = 2 (k - 1) (n + p - k + 1) + (n + p - k) + 2 (k - 1) (n + p - k) \text{ opérations.}$$

$$N_k = - 4 k^2 + (4n + 4 p + 5) k - 3n - 3 p - 2$$

L'obtention de la solution nécessite « n » éliminations du même type que la k° . Par conséquent, le nombre « N » d'opérations à effectuer pour obtenir la solution, sera :

$$N = \sum_{k=1}^{n} [- 4 k^2 + (4n + 4 p + 5) k - 3 n - 3 p - 2]$$

$$\boxed{N = \frac{n}{6} [4 n^2 - 3 n + 12 p n - 6 p - 1]}$$

Il faut noter que ce résultat est le même que celui calculé pour la méthode de Gauss (voir paragraphe *1-2-3*).

2 - 1 - 3 PROGRAMMATION

Si nous enregistrons initialement toutes les données, nous aurons un programme dont les possibilités seront comparables à celles de la méthode classique de JORDAN ou de GAUSS. Un tel programme ne se justifie pas. Au contraire, si nous exploitons la propriété de la méthode de JORDAN Progressive selon laquelle les équations sont introduites une à une, notre programme pourra traiter des systèmes linéaires d'ordre élevé. Nous nous sommes limités à l'étude de ce dernier cas, ce qui ne permet pas le choix du pivot maximal dans toute la matrice.

2-1-3-1 STRUCTURE du PROGRAMME

En prenant le pivot sur la diagonale principale de la matrice, le programme de la méthode de JORDAN Progressive se décompose en unesérie de sous-programmes.

- Sous-Programme de lecture d'une équation.
- Sous-Programme d'élimination des (k-1) premières inconnues de la k° équation.
- Sous-Programme de division de l'équation pivot par le pivot.
- Sous-Programme d'élimination de la k° inconnue.

A chaque étape du calcul correspond un sous-programme particulier. Finalement un programme de liaison organise l'enchaînement des calculs.

2-1-3-2 CAS d'un PIVOT NUL ou INFERIEUR à une LIMITE FIXEE

Le pivot n'étant pas choisi, mais pris obligatoirement sur la diagonale principale, il y a lieu de prévoir le cas où il est nul ou simplement inférieur à une limite fixée par l'utilisateur. A cet effet, nous avons prévu :

- $(n + p)$ mémoires de travail qui contiennent la dernière équation.
- un sous-programme pour perforer la dernière équation introduite lorsqu'elle ne peut pas fournir de pivot. Ce même sous-programme sera utilisé à la fin des calculs pour perforer les résultats définitifs.

Par conséquent, si après avoir éliminé les $(k - 1)$ premières inconnues dans la k^o équation, le coefficient de la k^o inconnue devient nul ou inférieur à la limite fixée, nous perforons la k^o équation sous sa forme initiale. Nous passons ensuite au traitement de la $(k+1)^o$ équation. Si elle peut fournir un pivot pour l'élimination de la k^o inconnue, les calculs se poursuivent normalement. Si au contraire elle ne peut pas en fournir, nous la perforons et passons à l'équation suivante et ainsi de suite.... S'il est impossible de trouver un pivot pour éliminer la k^o inconnue, il y a «arrêt». La matrice du système linéaire à résoudre est singulière. (voir paragraphe *1-1-1-2-2*).

Remarques

Toutes les équations perforées doivent être placées à la suite des données initiales dans l'unité de lecture. Elles seront traitées à un stade ultérieur du calcul.

Dans le cas où toutes les données sont initialement enregistrées en mémoire, les opérations précédentes sont remplacées par des permutations d'équations.

2-1-3-3 ENCOMBREMENT des DONNEES et des RESULTATS INTERMEDIAIRES

De cette façon, le nombre de mémoires utilisées pour les données et les résultats intermédiaires, au cours du traitement de la k^o équation, se décompose comme suit :

- $(n + p)$ mémoires pour conserver la k^o équation sous sa forme initiale.
- $(k - 1)(n - k + 1 + p)$ mémoires pour les $(k - 1)$ premieres équations.
- $(n - k + 1 + p)$ mémoires pour les formes successives de la k^o équation.

Ce qui nous donne au total :

$$n + p + k(n - k + 1 + p) \text{ mémoires.}$$

Le maximum de cette quantité est obtenu pour

$$2k = n + 1 + p$$

et vaut :

$$\frac{(n + p)(n + p + 6) + 1}{4}$$

Par conséquent, pour résoudre un système linéaire d'ordre n avec p séries de seconds membres, il faut disposer d'un nombre de mémoires inférieur ou au maximum égal à :

$$\frac{(n + p)(n + p + 6) + 1}{4}$$

Notons qu'en enregistrant initialement toutes les données, il faut obligatoirement :

$$(n+p)(n+1)+n \quad \text{mémoires.}$$

La méthode de JORDAN Progressive permet donc de diviser approximativement par 4 ce nombre de mémoires nécessaires.

2-1-3-4 MODES de CALCUL UTILISES

Nous avons programmé la méthode de JORDAN Progressive en virgule flottante type «FLAIR», simple et double précision. Le passage d'un de ces programmes à l'autre nécessite la manipulation de nombres occupant deux mémoires (partie gauche et partie droite du nombre en double précision) et l'emploi d'une arithmétique convenable. Les nombres en double précision occupent deux blocs de mémoires : un premier pour les parties gauches et un second pour les parties droites. Il s'agit de la même disposition que celle adoptée pour la méthode classique de JORDAN.

2 - 1 - 4 APPLICATION à la RESOLUTION des SYSTEMES LINEAIRES à MATRICE-BANDE.

La méthode de JORDAN Progressive nous a paru intéressante pour la résolution des systèmes linéaires à matrice-bande. Par «matrice-bande» nous entendons une matrice carrée, dans laquelle tous les termes situés en dehors de la diagonale principale et d'un même nombre de diagonales parallèles de part et d'autre de la diagonale principale, sont nuls. Nous désignons par «m» le nombre de diagonales constituant la bande et qui portent des termes à priori non nuls. Le nombre «m» est un nombre obligatoirement impair.

Nous avons écrit un sous-programme supposant les données (n équations) initialement enregistrées. Seuls les termes de la bande, à priori non nuls, et les seconds membres sont conservés en mémoires. Ainsi pour un système linéaire d'ordre «n», à matrice-bande caractérisée par «m», et avec p séries de seconds membres, le nombre de mémoires nécessaires pour enregistrer en simple précision les coefficients à priori non nuls, est égal à :

$$(p+m)\,n - \frac{m^2-1}{4}$$

Tout ce que nous avons vu concernant la méthode de JORDAN Progressive est toujours valable. Il s'agit seulement d'en faire l'adaptation au cas des systèmes linéaires à matrice-bande. Le mode opératoire reste pratiquement identique à celui exposé pour le cas général. Il n'en diffère que par les quelques points suivants :

- seuls les coefficients à priori non nuls étant enregistrés, toutes les opérations correspondant à des coefficients nuls ont été supprimées. De cette façon, lors du traitement de la k^o équation :
  - il n'y aura jamais plus de $\frac{m-1}{2}$ inconnues à éliminer dans cette équation.
  - la division de l'équation pivot par le pivot intéressera au maximum $(\frac{m-1}{2}+p)$ termes.
  - par contre, l'élimination de la k^o inconnue portera toujours sur les $(k-1)$ premières équations.
- le fait de permuter des équations détruit automatiquement la structure d'une matrice-bande qui devient une matrice quelconque. C'est pourquoi la découverte d'un pivot nul (choisi sur la diagonale principale) entraînera un arrêt. Il ne sera plus possible de poursuivre les calculs.

En prenant le pivot sur la diagonale principale, on démontre facilement que le nombre de coefficients non-nuls ou différents de l'unité, qu'il faut conserver en mémoire, n'augmente pas au cours de la résolution d'un système linéaire à matrice-bande. Au contraire, ce nombre diminue constamment à partir de sa valeur initiale :

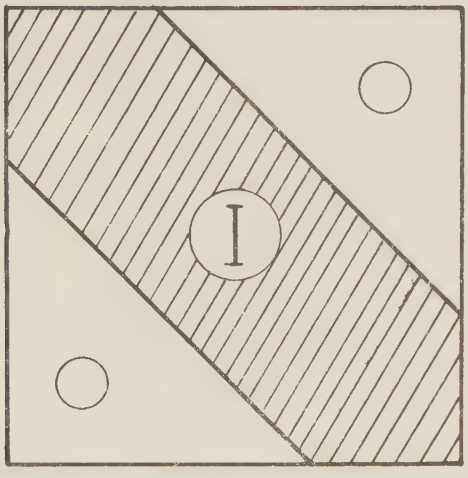
$$(m + p)\,n - \frac{m^2 - 1}{4}$$

pour devenir égal à «p*n*» à la fin des calculs. Cette dernière propriété est fondamentale et explique à elle seule tout l'intérêt que présente la méthode de JORDAN Progressive pour la résolution des systèmes linéaires à matrice-bande. Le nombre «p *n*» correspond aux «p» séries de «*n*» racines associées aux séries initiales de seconds membres. Il est à noter que ces «p *n*» seconds membres restent quelconques pendant tous les calculs, et sont de ce fait constamment conservés en mémoire. Seule la forme de la matrice-bande évolue au cours des calculs.

2-1-4-1 FORMES SUCCESSIVES de la MATRICE-BANDE

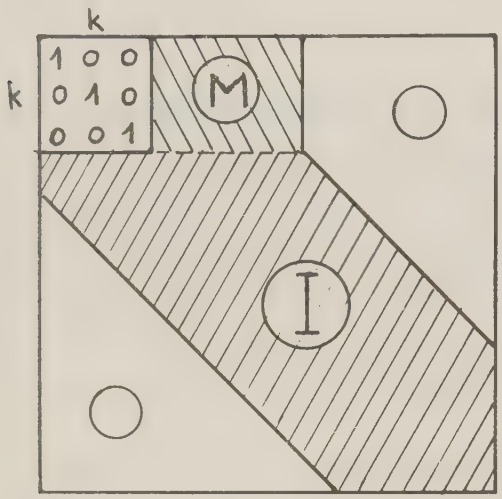
Pour avoir une idée de cette évolution, représentons schématiquement les formes successives prises par la matrice-bande. Les zones hachurées représenteront les coefficients à priori non nuls. La référence (I) indiquera qu'il s'agit de coefficients ayant encore leur valeur initiale (donnée). La référence (M) indiquera qu'il s'agit de coefficients ayant participé aux calculs.

2-1-4-1-1- Forme initiale de la matrice-bande.

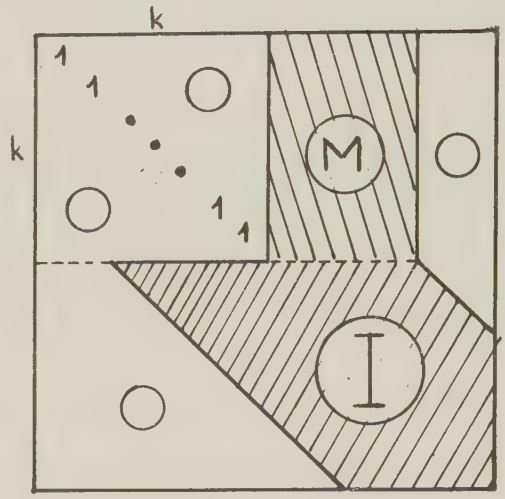


2-1-4-1-2 *Formes de la matrice bande après k éliminations*

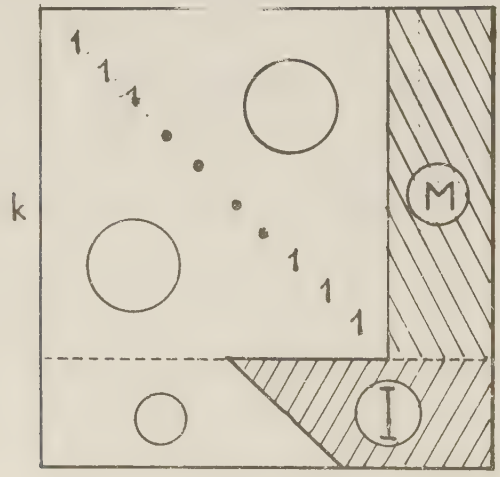
2-1-4-1-2-1 $k \leq \frac{m-1}{2}$



2-1-4-1-2-2 $\frac{m-1}{2} < k < n - \frac{m-1}{2}$



2-1-4-1-2-3 $n - \frac{m-1}{2} \leq k < n$



La matrice unité, qui occupe le coin supérieur gauche de la matrice-bande, voit son rang augmenter au fur et à mesure de l'avancement des calculs. La bande verticale composée de coefficients participant aux calculs (M) se déplace vers la droite tout en conservant une largeur constante, jusqu'à atteindre la limite de droite de la matrice-bande. Elle se réduit alors progressivement, et lorsque $k = n$, il nous reste une matrice unité d'ordre n. La résolution du système linéaire à matrice-bande est terminée, les derniers seconds membres calculés constituent les solutions recherchées.

2-1-4-2 NOMBRE d'OPERATIONS ARITHMETIQUES

En tenant compte du fait que seuls les coefficients à priori non-nuls sont conservés en mémoire et que les opérations correspondant à des coefficients nuls sont supprimées, le nombre d'opérations arithmétiques nécessaires à l'obtention de la solution d'un système linéaire à matrice-bande, à l'aide de la méthode de JORDAN Progressive, est très long à calculer. Nous nous contenterons de donner le résultat que nous avons vérifié en machine :

$$\left(\frac{m-1}{2}+p\right) n^2+ (m-1)\left(p+\frac{m-3}{4}\right) n - \frac{(m-1)^3}{4} - \frac{(m^2-1)(6p-m)}{24}$$

2-1-4-3 MODES de CALCUL UTILISES

La programmation de la méthode de JORDAN Progressive adaptée à la résolution des systèmes linéaires à matrice-bande, peut être envisagée en virgule flottante type «FLAIR», en simple ou en double précision. Les problèmes posés par le passage de la simple à la double précision sont absolument identiques à ceux rencontrés pour les méthodes de résolution de systèmes linéaires à matrice quelconque.

REMARQUE

Nous trouverons en annexe (Annexe I, 5 et 6), l'étude détaillée des programmes en virgule flottante simple précision relatifs à la méthode de JORDAN Progressive appliquée aux systèmes linéaires soit à matrice quelconque, soit à matrice-bande.

2-1-5 RESULTATS

2-1-5-1 CAS GENERAL

Les tableaux suivants nous donnent les limites d'utilisation, les temps de calcul etle nombre d'opérations arithmétiques pour la méthode de JORDAN Progressive programmée en virgule flottante simple et double précisions.

Possibilités Maxima	Méthode de JORDAN Progressive programmée en Virgule Flottante	
	Simple Précision	Double Précision
n_{MAX}	72	47
p_{MAX}	1	1

Remarque : La valeur p_{MAX} représente la valeur maximum permise pour p lorsque nous prenons n égal à n_{MAX}.

Ordre du système linéaire étudié ; nous avons toujours : p = 1.	Méthode de JORDAN Progressive programmée en Virgule Flottante SIMPLE PRECISION	
	Temps de calcul	Nombre d'Opérations
$n = 6$	21 secondes	191
12	2 mn. 07 s.	1354
18	6 mn. 32 s.	4353
24	14 mn. 40 s.	10 052
30	27 mn. 45 s.	19 315
36	47 mn. 05 s.	33 006

Ordre du système linéaire étudié : nous avons toujours : p = 1.	Méthode de JORDAN Progressive programmée en Virgule Flottante DOUBLE PRECISION	
	Temps de calcul	Nombre d'opérations
6	1 mn. 02 s.	191
12	6 mn. 35 s.	1354
18	20 mn.	4353
24	45 mn.	10 052
30		19 315
36		33 006

Comme nous l'avons déjà fait remarquer, les nombres d'opérations arithmétiques sont égaux à ceux nécessaires avec la méthode de GAUSS. Les temps de calcul sont du même ordre de grandeur que ceux mesurés pour la méthode de GAUSS sans choix du pivot (c'est-à-dire qu'il est pris obligatoirement sur la diagonale principale).

La précision des résultats obtenus en simple ou double précision, est fonction de l' ordre dans lequel se présentent les équations. Cette constatation n'est pas pour surprendre, car elle est le propre de toutes les méthodes d'élimination pour lesquelles aucun choix de pivot n'est effectué. En calculant en simple précision, avec un système linéaire d'ordre 36, [Annexe II, 4] dont toutes les racines sont inférieures à 3,7 en valeur absolue, nous avons détecté une erreur absolue maximale égale à 7.10^{-7} quand les équations étaient classées de façon à obtenir un système linéaire à diagonale principale prédominante. Cette erreur absolue maximale est devenue égale à 2.10^{-4} avec une autre disposition des équations pour laquelle le système linéaire n'était plus à diagonale principale prédominante.

Il est donc difficile de conclure quant à la précision des résultats obtenus par la méthode de JORDAN Progressive programmée en simple ou double précision. Comme pour tous nos programmes en virgule flottante simple précision, nous détectons la perte maximale de précision rencontrée au cours de chaque résolution. [Annexe I, 5] .. Signalons encore que si nous considérons un système linéaire quelconque, et si nous le résolvons par la méthode de GAUSS sans choix de pivot, puis par celle de JORDAN Progressif en conservant, dans les deux cas, les équations dans le même ordre, les résultats obtenus sont comparables. Les erreurs absolues maximales détectées sont du même ordre de grandeur.

Finalement, la méthode de JORDAN Progressive constitue un excellent prolongement de la méthode de GAUSS. Elle permet de traiter en simple ou double précision, les systèmes linéaires d'ordre élevé avec des temps de calcul et des nombres d'opérations arithmétiques optima.

Remarque

Le calcul de la matrice inverse n'est pas possible à l'aide de la méthode de Jordan classique, pour les systèmes linéaires d'ordre élevé ($n > 36$ en simple précision, $n > 24$ en double précision). La méthode de JORDAN Progressive pourra être utilisée. Nous serons obligés de recommencer plusieurs fois la résolution du même système linéaire en prenant à chaque fois le nombre maximum possible de colonnes de la matrice unité convenable.

Ainsi avec $n = 50$, nous avons en simple précision $p_{MAX} = 23$, ce qui entraîne qu'il faudra résoudre 3 fois le même système linéaire. La première fois, nous prendrons les 23 premières colonnes de la matrice unité d'ordre 50 comme seconds membres et nous obtiendrons les colonnes correspondantes de la matrice inverse, et ainsi de suite .....

2-1-5-2 CAS des SYSTEMES LINEAIRES à MATRICE-BANDE

Les tableaux suivants nous donnent les limites d'utilisation, les temps de calcul et les nombres d'opérations arithmétiques relatifs à la méthode de JORDAN Progressive appliquée à la résolution, en virgule flottante simple ou double précision, des systèmes linéaires à matrice-bande.

Limites d'utilisation	Méthode de JORDAN Progressive - appliquée à la résolution des systèmes linéaires à MATRICE-BANDE - programmée en VIRGULE FLOTTANTE :					
	Simple Précision			Double Précision		
	m = 3	m = 5	m = 7	m = 3	m = 5	m = 7
n_{MAX}	387	259	195	175	117	88
p_{MAX}	1	1	1	1	1	1

Remarque : La quantité p_{MAX} représente la valeur maximum permise pour p lorsque nous prenons n égal à n_{MAX}.

Caractéristiques du système linéaire			Méthode de JORDAN Progressive - appliquée à la résolution des systèmes linéaires à MATRICE-BANDE - programmée en Virgule Flottante :		
n	m	p	Simple Précision Temps de calcul	Double Précision Temps de Calcul	Nombre d'opérations
6	3	1	11 secondes	28 secondes	81
12	7	1	1 mn. 05 s.	3 mn. 25 s.	668
9	7	1	36 secondes	1 mn. 52 s.	380
89	3	1	27 mn.		16 017
179	3	1	106 mn.	5 h. 34 mn.	64 437

Les possibilités de la méthode de JORDAN Progressive appliquée à la résolution des systèmes linéaires à MATRICE-BANDE, sont très vastes. Elles pourraient être améliorées en travaillant non plus sur les données préenregistrées, mais en lisant les équations une à une comme dans le cas d'une matrice quelconque.

Les temps de calcul et les nombres d'opérations sont notablement réduits par rapport au programme général. Par exemple, pour un système linéaire d'ordre 12 [Annexe II, 7] dont la matrice-bande est caractérisée par m = 7, et pour une seule série de seconds membres, les gains sont de 49% pour le nombre d'opérations et de 51% pour le temps de calcul, par rapport au programme relatif aux matrices quelconques.

La précision des résultats n'est plus sous la dépendance de l'ordre des équations, comme dans le cas d'une matrice quelconque. En effet, dans un système linéaire à matrice-bande, chaque équation a une place parfaitement déterminée. Toute modification détruit la nature du système, la MATRICE-BANDE devient quelconque. D'autre part, les systèmes linéaires à matrice-bande étant souvent issus d'applications des équations aux différences, la diagonale principale est généralement prédominante. C'est le cas, par exemple, de la résolution d'équations différentielles avec conditions aux limites. Les systèmes linéaires à MATRICE-BANDE sont donc traités dans les meilleures conditions possibles. Les résultats obtenus seront généralement précis. Pour un système d'ordre 12, [Annexe II, 7] dont la matrice-bande est caractérisée par m = 7, nous avons obtenu les racines, toutes positives et inférieures à 1, 3, avec une erreur absolue maximale de $7 . 10^{-7}$ en simple précision et de $3 . 10^{-17}$ en double précision. La perte maximale de précision est de 2 chiffres au cours de la 5° élimination. Il ne nous est malheureusement pas possible d'évaluer la précision des résultats calculés pour les gros systèmes d'ordre 89 et 179. En effet, ils ont été obtenus en résolvant le problème aux limites

$$\begin{cases} y'' + y = 2 \sin x \\ y(0°) = y(90°) = 0 \end{cases}$$

et en prenant comme pas soit 1°, soit 30'. A la fin des calculs, la solution obtenue est entachée par rapport à la solution exacte $-x \cos x$ de l'erreur due à la méthode utilisée pour la résolution de l'équation différentielle, augmentée des erreurs de calculs dues en particulier à la résolution du système linéaire correspondant.

La méthode de JORDAN Progressive est donc bien adaptée à la résolution des systèmes linéaires à MATRICE-BANDE. En n'enregistrant en mémoire que les coefficients à priori non nuls, elle permet le traitement de systèmes d'ordre élevé. En limitant les calculs à ces mêmes coefficients, elle entraîne une économie appréciable d'opérations arithmétiques et par conséquent une réduction importante des temps de calcul.

2 - 2 *METHODE de DECOUPAGE de la MATRICE*

2-2-1 PRINCIPE

Lorsqu'il s'agit de traiter des systèmes linéaires d'ordre élevé ($n > 36$) il devient impossible d'enregistrer simultanément en mémoire : les données ($n^2 + p\,n$ valeurs) et le programme de calcul. Dans ces conditions, nous pouvons utiliser la méthode de découpage de la matrice. Ce découpage peut être effectué de multiples façons et conduire à un nombre quelconque de blocs (sous-matrices). Nous nous contenterons ici de considérer un découpage de la matrice A du système en 4 blocs (cas minimum) et opèrerons de la façon suivante.

Soit à résoudre le système linéaire : $A\,X = B$

avec : A : matrice $n \times n$

X et B : matrices $n \times p$.

Pour cela, nous l'écrirons sous la forme :

$$\left[\begin{array}{c|c} A_1 & A_2 \\ \hline A_3 & A_4 \end{array}\right] \left[\begin{array}{c} X_1 \\ \hline X_2 \end{array}\right] = \left[\begin{array}{c} B_1 \\ \hline B_2 \end{array}\right]$$

où $A_1, A_2, A_3, A_4, X_1, X_2, B_1$, et B_2 représentent des blocs. La nature du découpage est déterminée par les calculs ultérieurs. Comme nous désirons décomposer le système linéaire en deux équations matricielles

$$A_1 X_1 + A_2 X_2 = B_1$$

$$A_3 X_1 + A_4 X_2 = B_2$$

et calculer à partir de la première, la valeur de X_1 en fonction de X_2 :

$$X_1 = A_1^{-1} [B_1 - A_2 X_2]$$

ceci implique que :

a) le découpage vertical de A et les découpages horizontaux de X et B soient tous identiques (décomposition du système linéaire en deux équations matricielles).

b) les découpages horizontal et vertical de A soient les mêmes (inversion de la sous-matrice A_1 pour le calcul de X_1 en fonction de X_2).

Ces conditions étant réalisables, les calculs précédents sont possibles. Et en portant la valeur de X_1 en fonction de X_2 dans la seconde équation matricielle, nous obtenons :

$$[A_4 - A_3 A_1^{-1} A_2] X_2 = B_2 - A_3 A_1^{-1} B_1$$

La résolution de ce système linéaire nous fournira la valeur de X_2. Puis en reportant ce résultat dans l'expression de X_1 en fonction de X_2, nous obtiendrons la valeur de X_1. La résolution du système linéaire initial est alors terminée.

2-2-2 *PROGRAMMATION*

La méthode de découpage de la matrice doit être programmée de façon à traiter des systèmes linéaires d'ordre le plus élevé possible, tout en restant d'une utilisation pratique. C'est-à-dire qu'il faudra éviter au maximum les manipulations de cartes une fois le calcul commencé.

Il convient donc de laisser la plus grande place possible pour les données et les résultats intermédiaires au détriment de l'encombrement du programme. Nous verrons comment ceci est réalisable grâce à l'indépendance des diverses étapes du calcul.

Il faut encore que les données enregistrées à chaque étape du calcul aient un encombrement minimum. Cet impératif sera utilisé pour déterminer le découpage optimum de la matrice A, en dehors des impératifs théoriques auxquels nous sommes obligés de nous astreindre.

2-2-2-1 CHAINE de CALCULS

Examinons maintenant la chaîne de calculs nécessaire à la résolution d'un système suivant la méthode exposée au paragraphe précédent :

Numéro du calcul	Quantité calculée	. Données nécessaires	Résultats Intermédiaires nécessaires.
1	A_1^{-1}	A_1	
2	$A_3 A_1^{-1}$	A_3	A_1^{-1}
3	$A_4 - A_3 A_1^{-1} A_2$	A_4 et A_2	$A_3 A_1^{-1}$
4	$B_2 - A_3 A_1^{-1} B_1$	B_2 et B_1	$A_3 A_1^{-1}$
5	Résolution du système linéaire donnant X_2		$A_4 - A_3 A_1^{-1} A_2$ et $B_2 - A_3 A_1^{-1} B_1$
6	$B_1 - A_2 X_2$	B_1 et A_2	X_2
7	$X_1 = A_1^{-1} [B_1 - A_2 X_2]$		A_1^{-1} et $B_1 - A_2 X_2$

Si nous caractérisons le découpage de la matrice A par l'ordre «k» du bloc carré A_1, nous pouvons à l'aide du tableau ci-dessus mettre en évidence les parties successives du programme de calcul :

1° Partie (Calcul N° 1) : Nous lisons le bloc A_1 et nous calculons son inverse à l'aide d'un sous-programme d'inversion de matrice utilisant la méthode de JORDAN. La matrice A_1^{-1} est à la fois conservée en mémoire et perforée.

Le nombre de mémoires nécessaires pour l'enregistrement des données et des résultats intermédiaires est alors : $k(k+2)$.

(Cf. notice d'utilisation du sous-programme de résolution de systèmes linéaires et d'inversion de matrices en virgule flottante simple précision. [Annexe I, 2]).

2° Partie (Calculs N° 2, 3 et 4) : Cette seconde partie se déroule de la façon suivante :

- Lecture de la matrice A_3 ligne par ligne, calcul du produit $A_3 A_1^{-1}$ et mise en mémoire de la matrice résultat. Ce calcul nécessite $[k^2 + k + k(n-k)]$ mémoires pour les données et les résultats. La matrice A_1^{-1} peut maintenant être détruite.

- Lecture de toute la matrice A_2, lecture de la matrice A_4 ligne par ligne, tout en calculant $A_4 - A_3 A_1^{-1} A_2$ et en perforant les résultats au fur et à mesure de leur obtention. Ce calcul nécessite $[(n-k)k + (n-k) + k(n-k)]$ mémoires pour les données et les résultats intermédiaires.

- Enfin, lecture de toute la matrice B_1, lecture de la matrice B_2 ligne par ligne, tout en calculant $B_2 - A_3 A_1^{-1} B_1$ et en perforant les résultats au fur et à mesure de leur obtention. Ce calcul nécessite

$[kp + p + k(n - k)]$ mémoires pour les données et résultats intermédiaires.

L'ensemble de ces trois calculs fait intervenir, en dehors du sous-programme d'Arithmétique en Virgule Flottante simple précision, trois autres sous-programmes:

a) pour la lecture et l'enregistrement d'une matrice : ligne par ligne ou entièrement.

b) pour les calculs matriciels de la forme :

$M_1 - M_2 M_3$ ou $M_2 M_3$ (M_1, M_2 et M_3 représentant des matrices).

c) pour la perforation des résultats.

Le nombre maximum de mémoires à prévoir pour le stockage des données et des résultats intermédiaires, sera égal à la plus grande des trois quantités :

$$k(n + 1)$$
$$(n - k)(2k + 1)$$
$$p(k + 1) + k(n - k)$$

3° Partie (Calcul N° 5) : Après lecture et enregistrement de la matrice du système et des seconds membres (perforés lors de la seconde partie), nous résolvons le système linéaire :

$$[A_4 - A_3 A_1^{-1} A_2] X_2 = [B_2 - A_3 A^{-1} B_1]$$

à l'aide du sous-programme de résolution de systèmes linéaires en virgule flottante simple précision, utilisant la méthode de JORDAN. Comme nous pouvons le vérifier dans la notice d'utilisation de ce sous-programme [Annexe I, 1], ce calcul nécessite :

$$[(n - k + p)(n - k + 1) + (n - k)]$$

mémoires pour les données et les résultats.

4° Partie (Calculs N° 6 et 7) : A la fin de la 3° partie, la matrice X_2 a pris la place des « p » premières colonnes de la matrice du système linéaire dont elle est la solution (Cf la notice d'utilisation du sous-programme de résolution de système linéaire [Annexe I, 1]), par conséquent elle neutralise $[(n-k)(n - k+ p) - n + k]$ mémoires.

Lors de cette dernière partie, nous aurons successivement :

- Lecture de toute la matrice B_1, lecture de la matrice A_2 ligne par ligne tout en calculant $B_1 - A_2 X_2$ et en conservant la matrice résultat en mémoire. Ce premier calcul nécessite $[(n - k)(n - k + p) - n + k + kp + n - k]$ mémoires pour les données et les résultats.

- Finalement, lecture de la matrice A_1^{-1} ligne par ligne tout en calculant $A_1^{-1} [B_1 - A_2 X_2] = X_1$ et en conservant la matrice résultat en mémoire. Ce dernier calcul nécessite $[(n - k)(n - k+p) - n + k + 2kp + k]$ mémoires pour les données et les résultats, étant donné que l'on conserve X_2 en mémoires. On termine alors la résolution en perforant les matrices X_1 et X_2.

Cette dernière partie du calcul fait intervenir les sous-programmes comparables à ceux utilisés lors de la deuxième partie. Et elle nécessite pour le stockage des données et des résultats intermédiaires ou définitifs, un nombre maximum de mémoires qui sera égal à la plus grande des deux quantités :

$$(n - k)^2 + pn$$
$$(n - k)^2 + (n - k)(p - 1) + k(2p + 1)$$

2-2-2-2 DETERMINATION de la VALEUR OPTIMUM de k

Après cet exposé, il s'agit de déterminer la valeur optimum de k. L'étude comparative des encombrements relevés pour chaque partie du programme conduit aux résultats suivants :

1° - La valeur de k doit être choisie égale à la partie entière de $\frac{n}{2}$, pour avoir un encombrement minimum et ceci quel que soit p inférieur à $\frac{(n+2)}{6}$. Nous avons alors l'encombrement maximum qui nous est donné par :

$$(n-k)(2k+1)$$

2° - La valeur choisie pour k afin d'avoir un encombrement minimum doit être supérieure à la partie entière de $\frac{n}{2}$, et devient fonction de p si cette dernière quantité est supérieure à $\frac{(n+2)}{6}$.

L'encombrement maximum nous est alors donné par : $(n-k)^2 + (n-k)(p-1) + k(2p+1)$.

Si nous faisons intervenir le nombre d'opérations élémentaires nécessaires pour résoudre un système linéaire par la méthode de découpage de la matrice, nous constatons quece nombre, qui est égal à :

$$k^3 + (n - \frac{5}{2})\,k^2 + (-n^2 + 3n + p - \frac{1}{2})\,k + n^3 - \frac{3n^2}{2} + 2pn^2 - pn + \frac{n}{2}$$

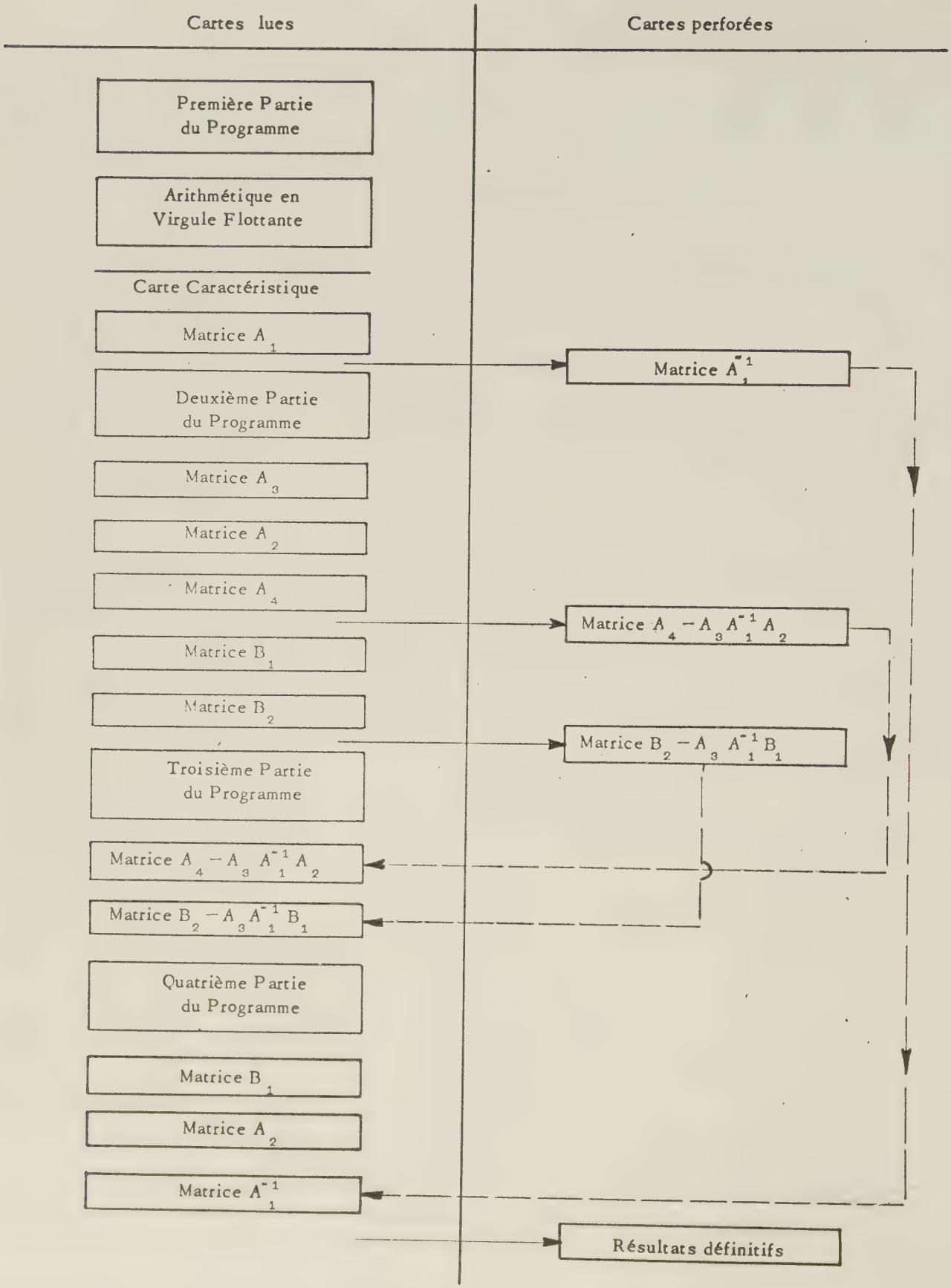
est minimum pour k voisin de $\frac{n-1}{3}$.

La valeur de p étant généralement faible, et n'entrant en ligne de compte que lorsque nous approchons des limites d'utilisation du programme, nous avons adopté une solution générale qui consiste à prendre la valeur de k égale à la partie entière de $\frac{n}{2}$. De cette façon, tout en obtenant des limites d'utilisation convenables, le nombre d'opérations élémentaires nécessaires pour résoudre un système linéaire reste le plus voisin possible de la valeur minimum.

2-2-2-3 DEROULEMENT des CALCULS

Le tableau suivant donnera une idée du déroulement des calculs de résolution d'un système linéaire d'ordre n avec p séries de seconds membres ; ainsi que la façon de présenter les données à la machine :

(voir page suivante)


Cartes lues
Cartes perforées
Première Partie du Programme
Arithmétique en Virgule Flottante
Carte Caractéristique
Matrice A_1
Deuxième Partie du Programme
Matrice A_3
Matrice A_2
Matrice A_4
Matrice B_1
Matrice B_2
Troisième Partie du Programme
Matrice $A_4 - A_3 A_1^{-1} A_2$
Matrice $B_2 - A_3 A_1^{-1} B_1$
Quatrième Partie du Programme
Matrice B_1
Matrice A_2
Matrice A_1^{-1}
Matrice A_1^{-1}
Matrice $A_4 - A_3 A_1^{-1} A_2$
Matrice $B_2 - A_3 A_1^{-1} B_1$
Résultats définitifs

Chaque partie du programme, ainsi que l'Arithmétique en Virgule Flottante, se présente sous forme de cartes «chargements» 6 mots. Un programme de chargement spécial (type chargement PASO) permet de contrôler le parfait enregistrement du programme et de démarrer les calculs. Les différentes parties du programme viennent s'enregistrer simultanément dans une même zone du tambour réservée à cet effet (mémoires 1400 à 1799). L'arithmétique en virgule flottante occupe en permanence les mémoires 1800 à 1999. Il y a donc superposition des différentes parties du programme. Les mémoires 0001 à 1399 sont réservées pour les données et les résultats intermédiaires.

La carte caractéristique contient les valeurs de n, p et k (k égale la partie entière de $\frac{n}{2}$).

Les paquets de cartes correspondant aux diverses sous-matrices données ou perforées, sont munis d'indicatifs précis : contrôlés par le programme lors de leur enregistrement, afin d'éviter tout calcul portant sur des valeurs fausses. Toute fausse manipulation de cartes conduit à un arrêt programmé.

2-2-3 *RESULTATS GENERAUX*

2-2-3-1 LIMITES d'UTILISATION du PROGRAMME de DECOUPAGE de la MATRICE

Nous avons vu comment déterminer la valeur de k, de façon à envisager le traitement de systèmes linéaires d'ordre le plus élevé possible. Nous avons constaté qu'il fallait prendre k égal à la partie entière de $\frac{n}{2}$, ce qui conduit à un encombrement au plus égal à $(n - k)(2k + 1)$ mémoires, à condition de prendre p inférieur ou égal à la partie entière de $\frac{n+2}{6}$. On verra facilement que pour n pair ou impair, l'expression $(n - k)(2k + 1)$ s'écrit $\frac{n(n+1)}{2}$. Nous obtiendrons alors la valeur maximum de n en résolvant l'équation $\frac{n(n+1)}{2} = 1400$.

puisque nous disposons au maximum de 1400 mémoires pour les données et les résultats intermédiaires. Nous obtenons de cette façon $n_{MAX} = 52$ avec p au plus égal à 9. Ceci constitue la limite d'utilisation du programme de résolution de systèmes linéaires par la méthode de découpage de la matrice.

2-2-3-2 DETERMINATION de la VALEUR MAXIMUM de p pour une VALEUR de n DONNEE.

Etudions maintenant le problème inverse que rencontre tout utilisateur du programme de résolution de systèmes linéaires par la méthode de découpage de la matrice. Etant donné une valeur de n, inférieure à 52, quelles valeurs de p pouvons-nous prendre, sachant que nous disposons au maximum de 1400 mémoires ? Il faudra que tous les encombrements fonction de p, rencontrés au paragraphe «Programmation», soient inférieurs à 1400. Avec la valeur de k égale à la partie entière de $\frac{n}{2}$, on constate qu'il suffit de vérifier l'inéquation suivante :

$$(n - k)^2 + (n - k)(p + 1) + k(2p + 1) \leq 1400$$

2-2-3-3 COMPARAISON de la METHODE du DECOUPAGE de la MATRICE avec les AUTRES METHODES DIRECTES

La méthode de découpage de la matrice a pour but d'envisager le traitement des systèmes linéaires d'ordre élevé. Elle constitue donc un mode de prolongement de la méthode classique de JORDAN ou de

GAUSS. Il semble donc intéressant de comparer ces deux méthodes entre elles, ainsi qu'avec la méthode de JORDAN Progressive, qui est un autre mode de prolongement de la méthode classique de JORDAN ou de GAUSS. Ces comparaisons peuvent se résumer à l'aide des tableaux suivants :

Méthodes programmées en Virgule Flottante Simple Précision	Limites d'utilisation
JORDAN	$n = 37$ $p = 2$
JORDAN Progressive	$n = 72$ $p = 1$
Découpage de la Matrice	$n = 52$ $p = 9$

METHODES	Temps de calcul pour $p = 1$ et différentes valeurs de n			Nombre d'opérations pour $p = 1$ et différentes valeurs de n		
	$n = 6$	$n = 12$	$n = 36$	$n = 6$	$n = 12$	$n = 36$
JORDAN	32 s..	3 mn 28 s	72 mn 16 s	231	1794	47286
JORDAN Progressive	21 s.	2 mn 07 s	47 mn 05 s	191	1354	33006
Découpage de la Matrice	36 s.	2 mn 58 s	61 mn 15 s	237	1707	42597

Pour la méthode du découpage de la matrice, les temps de calcul sont donnés par excès. En effet il est impossible de les mesurer exactement ; à certains moments, le programme lit, calcule et perfore simultanément.

2-2-3-4 PRECISION des RESULTATS

En ce qui concerne la précision des résultats obtenus, nous donnerons les erreurs absolues maximales détectées lors de la résolution d'un système linéaire d'ordre 36, [Annexe II, 4] dont les racines exactes sont connues, (la plus grande en valeur absolue est égale à 3,6 et la plus petite à 0,1) par les trois méthodes considérées précédemment.

Méthodes utilisées	Erreurs absolues maximales
JORDAN	$7 . 10^{-7}$
JORDAN Progressive	$7 . 10^{-7}$ à $1 . 10^{-4}$
Découpage de la Matrice	$7,2 . 10^{-5}$

Dans le cas de la méthode de JORDAN Progressive, l'erreur absolue maximum dépend de l'ordre dans lequel on place les différentes équations du système linéaire. Ce qui s'explique par le fait que les pivots sont pris obligatoirement sur la diagonale principale.

Il ne faut pas attacher trop de signification aux résultats ci-dessus. Néanmoins, l'expérience montre que la méthode du découpage de la matrice fournit des résultats bien moins précis que la méthode classique de JORDAN. Les résultats présentent en outre la particularité suivante : les racines obtenues en premier lieu (elles sont représentées par la matrice X_2, aux paragraphes précédents) sont beaucoup plus précises que celles obtenues en dernier lieu (elles sont représentées par la matrice X_1). A titre d'exemple, signalons que pour le système linéaire d'ordre 36 déjà considéré, l'erreur absolue maximum sur les coefficients de X_1 est de $7,2 . 10^{-5}$ et de $4 . 10^{-6}$ sur les coefficients de X_2, soit dix-huit fois plus faible.

Il serait possible d'apporter deux améliorations d'ordres différents, à la méthode du découpage de la matrice :

- la première amélioration concerne la précision des résultats, et consisterait à programmer la méthode en virgule flottante double précision. Il ne semble pas que cette solution soit entièrement satisfaisante. En effet, il ne faut pas espérer pouvoir traiter des systèmes linéaires d'ordre supérieur à 34 ; alors que la méthode classique de JORDAN en virgule flottante double précision permet déjà d'aller jusqu'à l'ordre 24, et celle de JORDAN Progressive, jusqu'à l'ordre 47.

- la seconde amélioration concerne l'ordre maximum des systèmes linéaires qu'il est possible de traiter. On peut espérer des résultats intéressants en effectuant des découpages plus importants (découpage de la matrice A du système en 9 sous-matrices par exemple). Mais en contre-partie, on compliquera sérieusement le programme ainsi que les manipulations de cartes en cours de calcul.

En conclusion, la méthode de découpage de la matrice ne présente aucun avantage sérieux sur la méthode de JORDAN Progressive. Par ailleurs, on lui reprochera deux gros défauts :

- la précision qui n'est pas du même ordre de grandeur pour toutes les racines d'un même système linéaire.

- les manipulations de cartes en cours de calcul, bien que réduites au minimum, ainsi que la nécessité d'intercaler les données entre les diverses parties du programme.

REMARQUES GENERALES

3 - 1 COMPARAISON des METHODES de GAUSS et de JORDAN

La méthode de GAUSS (en simple ou double précision suivant le cas) avec choix du pivot maximal dans toute la matrice est la meilleure méthode directe dont nous disposons actuellement pour la résolution des systèmes linéaires. Les résultats obtenus sont comparables à ceux fournis par la méthode de JORDAN, mais les temps de calcul sont plus faibles et ceci quel que soit le nombre «p» de séries de seconds membres. Avec p = 1, et en calculant en virgule flottante simple précision, on constate des gains de temps allant de 15 à 30% lorsque n varie de 6 à 36. Mettons en évidence cet avantage de la méthode de GAUSS sur celle de JORDAN, en calculant la différence $N_G - N_J$ (N_G : nombre d'opérations arithmétiques nécessaires pour l'obtention de la solution d'un système linéaire par la méthode de GAUSS, (Cf paragraphe *1-2-3*) et N_J : même signification pour la méthode JORDAN, (Cf paragraphe *1-1-3*).

$$N_G - N_J = n\left[\frac{2n^2}{3} + (2p - \frac{1}{2})n - (p + \frac{1}{6})\right] - \frac{n}{2}(2n - 1)(n + 2p - 1) = -\frac{n}{3}(n - 1)(n - 2)$$

Cette quantité est indépendante de p et est négative pour $n > 2$.

Par contre, la méthode de GAUSS ne présente plus les mêmes avantages lorsqu'il s'agit d'effectuer une inversion de matrice. En effet, si l'on calcule le nombre d'opérations arithmétiques nécessaires pour l'obtention d'une matrice inverse d'ordre n à l'aide des méthodes de GAUSS ou de JORDAN, on obtient le même résultat :

$$n^2(2n - 1) .$$

Ceci suppose que dans les deux cas, la nouvelle colonne de la matrice inverse introduite à chaque élimination est mise à la place des coefficients de l'inconnue éliminée. Si au contraire, on suppose l'enregistrement initial de la matrice unité d'ordre n (ce qui immobilise inutilement n^2 mémoires), le nombre d'opérations arithmétiques sera obtenu en faisant $p = n$ dans les quantités N_G et N_J données précédemment. Et dans ce cas, la méthode de GAUSS est plus rapide que celle de JORDAN.

3 - 2 INFLUENCE de l'ARRONDI du RESULTAT de CHAQUE OPERATION ARITHMETIQUE en VIRGULE FLOTTANTE sur les RESULTATS DEFINITIFS

Les méthodes de résolution de systèmes linéaires et d'inversion de matrices ayant été étudiées sur l'ordinateur I.B.M. 650 Standard, nous avons utilisé pour les programmes en simple précision : une Arithmétique en Virgule Flottante. Pour étudier l'influence de l'arrondi du résultat de chaque opération arithmétique en virgule flottante simple précision, sur les résultats définitifs, nous avons calculé la solution d'un système linéaire d'ordre 18 dont les données se trouvent en annexe [Annexe II, 3] à l'aide de la méthode de GAUSS. Les résultats obtenus figurent dans les tableaux ci-dessous :

Méthode de GAUSS avec choix du pivot dans toute la matrice

$n = 18$; $p = 1$

Nombre d'opérations arithmétiques : 4353

RESULTATS

Racines	Valeurs exactes	Arithmétique Virgule Flottante avec arrondi	Arithmétique Virgule Flottante sans arrondi
x_1	0,1	0,099999986	0,099999901
x_2	1,1	1,1	1,0999996
x_3	2,1	2,1000001	2,0999990
x_4	3,1	3,1	3,1000012
x_5	0,2	0,2	0,19999999
x_6	1,2	1,2000002	1,1999991
x_7	2,2	2,2000001	2,2000001
x_8	3,2	3,2000001	3,1999994
x_9	0,3	0,30000002	0,29999992
x_{10}	1,3	1,3000002	1,3000003
x_{11}	2,3	2,3	2,2999995
x_{12}	3,3	3,3000002	3,2999997
x_{13}	0,4	0,39999995	0,39999990
x_{14}	1,4	1,4	1,3999998
x_{15}	2,4	2,4000001	2,3999994
x_{16}	3,4	3,4	3,4000006
x_{17}	0,5	0,5	0,49999991
x_{18}	1,5	1,5000001	1,4999991

L'importance de l'arrondi n'est donc pas négligeable. En effet, pour les résultats précédents, l'erreur maximale en valeur absolue passe de $2 . 10^{-7}$ avec l'arrondi à $12 . 10^{-7}$ lorsqu'on le supprime. Cet écart provient du fait qu'avec l'arrondi : l'erreur sur chaque résultat d'opération arithmétique est : soit par excès, soit par défaut, toujours inférieure ou au maximum égaleà 5 unités du même ordre que la première décimale négligée. Tandis que sans l'arrondi, l'erreur sur chaque résultat d'opération arithmétique est toujours, par défaut, inférieure ou au maximum égaleà 9 unités du même ordre que la première décimale négligée. A titre d'exemple, pour la résolution du système linéaire d'ordre 18, précédent : 213 1 opérations ont été arrondies par excès et 2222 par défaut.

3 - 3 SYSTEMES LINEAIRES MAL CONDITIONNES - CAS PARTICULIER des MATRICES de HILBERT .

Deux essais ont été effectués avec la matrice de HILBERT d'ordre 6, et à l'aide de la méthode de JORDAN Itérations.

En prenant cette matrice sous la forme :

$$S_6 = \left\| \frac{1}{i + j - 1} \right\|_{i,\, j \,=\, 1, \ldots\ldots, 6}$$

et en limitant chaque coefficient à ses huit premiers chiffres significatifs (le huitième étant convenablement arrondi), nous travaillons sur des données entachées d'erreurs. Dans ces conditions, les résultats obtenus pour la matrice inverse, après trois itérations, n'ont aucune signification, mais sont identiques à ceux obtenus en double précision et arrondis à 8 chiffres significatifs.

En opérant ensuite avec la même matrice multipliée par 11 ! , de façon à travailler sur des données exactes, tous les coefficients de la matrice inverse ont été obtenus exactement, après deux itérations. Les dernières corrections étaient toutes nulles.

3 - 4 PERTES de PRECISION DETECTEES au COURS de la RESOLUTION d'un SYSTEME LINEAIRE.

3 - 4 - 1 EXEMPLE N° 1

Nous avons pris comme exemple un système linéaire issu d'une méthode de moindres carrés. Il est d'ordre 6 et mal conditionné; nous trouverons les données correspondantes en [Annexe II, 5] .

En utilisant la méthode de JORDAN programmée en virgule flottante simple précision, nous avons détecté toutes les pertes de précision rencontrées lors de la résolution du système ci-dessus. Ceci nous a permis de construire un tableau, dans lequel nous avons porté le nombre de pertes de précision relatives à une élimination et à un nombre donné de chiffres significatifs perdus :

Eliminations	Nombre de chiffres significatifs perdus				
	1	2	3	4	5
1°	25	3	o	o	o
2°	10	5	o	2	2
3°	6	o	o	o	o
4°	4	4	1	o	o
5°	2	o	o	o	o
6°	1	o	o	o	o

Ainsi, au cours de la 2° élimination, nous avons perdu 2 fois 5 chiffres significatifs. Pour l'une de ces pertes, le calcul effectué était le suivant :

$$3,8547910 - \frac{4,5350120}{4,9448000} \times 4,2030760$$

Un système linéaire présentant de telles pertes de précision au début des calculs (2° élimination) est mal conditionné. Les résultats obtenus en virgule flottante simple précision fourniront seulement un ordre de grandeur des valeurs exactes, comme on peut le constater en se reportant au paragraphe «Résultats» relatif à la méthode de JORDAN (paragraphe 1-1-4-4).

Remarques - Tous les coefficients calculés par élimination au cours de la première élimination ont subi une perte de précision d'au moins 1 chiffre significatif.

On trouvera à la fin du paragraphe suivant (*3-4-2*) une rapide comparaison des exemples N° I et N° II.

3 - 4 - 2 EXEMPLE N° II

Il s'agit d'un système linéaire d'ordre 4 extrait d'une communication de Petear B. MADIC (Electrical Engineer, Institute «Boris Kidric», Belgrade, Yugoslavia) aux «Journées Internationales de Calcul Analogique» (Septembre 1955). Nous trouverons les données de ce système linéaire en annexe [Annexe II, 6].

En utilisant la même méthode qu'au paragraphe *3-4-1* et en opérant de la même façon, nous avons obtenu le tableau suivant dans lequel est porté le nombre de pertes de précision relatives à chacune des 4 éliminations et à un nombre donné de chiffres significatifs perdus.

Eliminations	Nombre de chiffres significatifs perdus			
	1	2	3	4
1	o	o	5	7
2	3	2	2	o
3	o	1	1	o
4	o	o	o	o

Remarques

Tous les coefficients calculés par élimination au cours de la première élimination ont subi une perte de précision d'au moins 3 chiffres significatifs. Les résultats obtenus par la méthode de JORDAN programmée en virgule flottante simple précision sont sans signification comme on peut le constater au paragraphe 1-3-4-2-2-1-2 .

Si l'on compare les pertes de précision détectées au cours de la résolution des systèmes linéaires des exemples N° I et N° II , on constate en particulier que les pertes maximales ont lieu pour l'exemple N° I : au cours de la 2ème élimination (2 pertes de 5 chiffres) et pour l'exemple N° II : au cours de la 1° élimination (7 pertes de 4 chiffres). Cette comparaison peut expliquer les comportements différents des deux systèmes lors de leur résolution soit par la méthode de JORDAN programmée en virgule flottante double précision, soit par la méthode de JORDAN Itérations (voir paragraphe 1-3-4-2-2-1).

La détection des pertes de précision permet de se faire une idée du conditionnement d'un système linéaire. Il est possible de prévoir ces pertes, sans être obligé de résoudre le système linéaire, en considérant la matrice des coefficients du système. Ainsi pour le système de l'exemple N° II, nous avons une matrice dont toutes les lignes prises deux à deux sont presque proportionnelles. Par exemple, si nous divisons les coefficients de la première ligne par leur homologue pris dans la 2° ligne, nous obtenons :

$$\frac{a_{11}}{a_{21}} = 0{,}71939581$$

$$\frac{a_{12}}{a_{22}} = 0{,}71939274$$

$$\frac{a_{13}}{a_{23}} = 0{,}71943138$$

$$\frac{a_{14}}{a_{24}} = 0{,}71944676$$

Le même phénomène existe pour la matrice du système linéaire de l'exemple N° II, mais il est moins prononcé

CONCLUSION

La résolution des systèmes linéaires, considérés sous leur forme la plus générale, se fera dans les meilleures conditions de rapidité et de précision, à l'aide des méthodes directes. Parmi celles-ci, les méthodes de GAUSS et de JORDAN seront le plus couramment utilisées. La méthode de GAUSS étant la plus rapide, sera généralement préférée à celle de JORDAN. Si le système linéaire à résoudre est « *mal conditionné* », nous aurons recours : soit aux mêmes méthodes que précédemment programmées en double précision ; soit à la Méthode de JORDAN-Itérations.

Le traitement des systèmes linéaires d'ordre élevé ($n > 36$ en simple précision) sera facilité par l'emploi de la méthode de JORDAN PROGRESSIVE calculant soit en simple, soit en double précision.

L'inversion d'une matrice, avec ou sans résolution d'un système linéaire, sera effectuée, sans enregistrement de la matrice unité, à l'aide de la méthode de JORDAN.

Le problème très particulier des systèmes linéaires à MATRICE-BANDE est bien résolu par l'emploi de la méthode de JORDAN PROGRESSIVE. En simple ou en double précision, les calculs sont rapides, les résultats précis et les possibilités très vastes.

Finalement, tous les problèmes se rapportant aux systèmes linéaires considérés sous leurs différentes formes possibles, sont susceptibles d'être résolus dans les meilleures conditions. La détection de la perte maximum de précision permet de juger du conditionnement d'un système, une fois sa résolution terminée. Il serait préférable de disposer d'un critère simple et rapide décelant les systèmes « *mal conditionnés* » sans avoir besoin de les résoudre. On pourrait alors utiliser immédiatement une méthode convenable permettant d'obtenir une bonne précision.

2ème PARTIE

ANNEXES

ANNEXE I

Programmes

pour

l'Ordinateur I.B.M. 650 Standard

L'étude de toute méthode de calcul commence par l'écriture d'un premier programme. En fonction des résultats obtenus, il sera progressivement amélioré, pour se présenter finalement sous une forme qui soit la meilleure possible. Cette évolution se fera plus ou moins rapidement et entraînera l'écriture d'un plus ou moins grand nombre de programmes intermédiaires.

L'étude desprincipales méthodes de résolution des systèmes linéaires, qui a fait l'objet de la première partie de cet exposé, a nécessité l'écriture d'une trentaine de programmes. Nous allons, dans cette seconde partie, étudier en détail les programmes les plus représentatifs de chacune des meilleures méthodes. Tous ces programmes calculent en Virgule Flottante du type *Flair* : soit en Simple Précision, soit en Double Précision. L'étude d'un programme comprend :

1°/ une notice d'utilisation en Simple Précision, et éventuellement une autre : en Double Précision, si le programme correspondant existe.

2°/ une note explicative de la Simple Précision qui consiste en l'étude de toutes les instructions.

Remarque - Nous ne donnons pas de note explicative de la Double Précision qui peut se déduire aisément de celle de la Simple Précision. En effet, le passage en Double Précision conserve la logique du programme en Simple Précision ; il faut seulement prévoir la manipulation de nombres occupant deux mémoires et remplir les conditions d'utilisation du Sous-Programme d'Arithmétique en Virgule Flottante Double Précision adopté.

Nous ne trouverons ici aucun programme relatif aux méthodes des GRADIENTS CONJUGUES et du DECOUPAGE de la MATRICE. L'utilisation de ces deux méthodes ne présente en effet aucun avantage pratique.

L'emploi de tous les programmes étudiés nécessite encore dans la pratique l'écriture d'un certain nombre de programmes de « servitude ». Par exemple :

- chargement d'une matrice quelconque
- chargement d'une MATRICE-BANDE
- perforation des résultats
- perforation d'une matrice

etc.....

Nous ne donnerons aucune précision sur ces programmes qui ne présentent pas de particularité intéressante. S'il est nécessaire, leur écriture peut se faire rapidement et sans grosse difficulté.

NOTATION

En dehors des notations propres à certains paragraphes, qui seront définies au moment de leur introduction, précisons quelques caractéristiques des « *Notes explicatives* » des programmes.

STRUCTURE d'une LIGNE

Une *Note Explicative* a pour but d'étudier toutes les instructions d'un programme. L'étude d'une seule instruction nécessite l'écriture d'une ligne que nous décomposons en quatre zones indiquées sur l'exemple ci-dessous :

Zone I	*Zone II*			*Zone III*	*Zone IV*	
(21)	0100	69	0103	0106	(32)	(0103) = - 12 3456 7890

- *Zone I*

Le nombre entre parenthèses placé dans cette zone sert de référence pour l'instruction correspondante. Ceci s'avère nécessaire chaque fois que nous rencontrons une instruction à laquelle nous sommes sûrs d'être renvoyés à un moment quelconque du programme, (par exemple : première instruction d'une boucle).

- *Zone II*

Nous y trouvons l'adresse de l'instruction à étudier et cette instruction en langage machine.

- *Zone III*

Elle n'est utilisée que dans le cas où l'instruction à exécuter immédiatement après l'instruction intéressée, n'est pas étudiée à la ligne suivante. Nous trouvons alors dans la zone III, un nombre entre parenthèses que nous rechercherons dans la zone I d'uneautre ligne, afin de poursuivre l'étude du programme. Citons comme exemple : le renvoi à la première instruction d'une boucle après un test de «Fin de boucle».

- *Zone IV*

Elle est réservée aux *Commentaires*. Elle comportera le plus souvent les contenus des mémoires de travail, ou toute autre indication nécessaire à la bonne compréhension du programme.

Pour représenter le contenu d'une mémoire, nous avons adopté la notation classique qui consiste à écrire l'adresse de la mémoire entre parenthèses. Par exemple :

(0103) = - 12 3456 7890

se lit : «le contenu de la mémoire d'adresse 0103 est égal à - 12 3456 7890».

REPRESENTATION d'un SOUS-PROGRAMME

Chaque fois que cela était possible, nous avons décomposé nos programmes en une série de sous-programmes et en un programme de liaison réalisant leur enchaînement.

Les sous-programmes étant étudiés séparement, nous avons convenu de schématiser chacun d'eux, dans le programme de liaison, à l'aide d'un rectangle à l'intérieur duquel nous plaçons :

- au centre : la raison d'être du sous-programme,

- dans le coin supérieur gauche : l'adresse de la mémoire contenant la première instruction du sous-programme,

- dans le coin inférieur droit : l'adresse de la mémoire contenant la dernière instruction du sous-programme (instruction de sortie).

Nous donnons au paragraphe suivant un exemple de cette représentation d'un sous-programme.

SOUS-PROGRAMME PARTICULIER : l'A. V. F. (Arithmétique en Virgule Flottante simple précision).

Dans tous les programmes qui suivent, les calculs sont effectués en Virgule Flottante, du type *Flair*, en Simple Précision à l'aide du Sous-Programme A.V.F.. Les conditions d'utilisation de ce Sous-Programme sont les suivantes :

- à l'entrée : nous devons charger en :

8003 : le premier facteur de l'opération arithmétique (dividende, multiplicande ....)

8002 : 00 0000 0000

8001 : une instruction de *Sortie*

1801 : le second facteur de l'opération arithmétique (diviseur, multiplicateur.....)

1808 : une constante d'orientation propre à chaque type d'opération (addition, soustraction, multiplication, division, etc....).

- à la sortie : les conditions sont les mêmes que pour un ordinateur I.B.M. 650 muni de la Virgule Flottante Automatique.

Par suite, chaque opération arithmétique en virgule flottante se traduit dans les *Notes Explicatives* de nos programmes par :

1°/ Une Séquence d'Entrée dans l'A.V.F. Le premier facteur de l'opération arithmétique étant déjà chargé en 8003, la séquence d'entrée dans l'A.V.F. se compose de cinq instructions que nous avons toujours encadrées. Elles ont pour but de charger convenablement les mémoires 1801 et 1808 ainsi que le distributeur 8001.

D'autre part nous indiquons toujours, en langage PASO, dans la zone *Commentaires* de l'instruction chargeant 1808 : le code Virgule Flottante Automatique correspondant à l'opération préparée (VAA, VSV, VMU, VDV ....).

2°/ Une représentation schématique du Sous-Programme A.V.F. Avec les notations définies au paragraphe précédent, nous aurons :

Remarque

Pour une étude complète du Sous-Programme A.V.F., nous renvoyons à la bibliographie [Annexe III, 12].

I, 1 - SOUS-PROGRAMMES de RESOLUTION de SYSTEMES LINEAIRES en SIMPLE PRECISION et en DOUBLE PRECISION. METHODE de JORDAN. (Références : J S P 1 et J D P 1)

I, 1-1 NOTICES d'UTILISATION

I, 1-1-1 SOUS-PROGRAMME en SIMPLE PRECISION

Objet : Etant donné la matrice et les vecteurs seconds membres d'un système linéaire, ce sous-programme calcule en virgule flottante *Flair* et en simple Précision, les vecteurs solutions.

Conditions d'utilisation : Si « n » désigne l'ordre de la matrice du système linéaire, et « p » le nombre de vecteurs seconds membres, la limite d'utilisation de ce sous-programme est :

$$(n+1)(n+p) + n \leqslant 1\,500$$

Ainsi, avec p = 1, nous avons : $n_{MAX} = 37$

La virgule flottante simulée étant du type *Flair*, on représentera :

1 par 51 1000 0000 +

Il faudra enregistrer l'A.V.F. (Arithmétique en virgule flottante simple précision) sur le tambour (mémoires occupées) : 1 800 à 1 999 sauf les mémoires d'entrée et de sortie standard de la bande 1 950).

Utilisation : Le système à résoudre doit être rangé ligne par ligne dans des mémoires consécutives du tambour. Chaque ligne se composant des n coefficients du premier membre et des p coefficients des seconds membres, c'est-à-dire d'une équation complète du système.

Le calcul utilise ($2n + p - 1$) mémoires auxiliaires pour les calculs et résultats intermédiaires. Ces ($2n + p - 1$) mémoires doivent se trouver en séquence, le bloc qu'elles occupent étant situé immédiatement après celui occupé par les données.

Si nous désignons par M l'adresse de la première mémoire du bloc où est rangé le système (ligne par ligne), l'utilisateur devra placer en :

8003 0 0 0 0 0 0 M +

8002 0 0 0 0 n n + p +

8001 une instruction de sortie

Après les calculs, les mémoires qui contenaient les p premières colonnes de la matrice donnée, contiennent les p vecteurs solutions. Nous pouvons très bien avoir $p > n$. La seule limite étant celle due à la formule du paragraphe précédent.

Exemple - Supposons : $n = 6$, $p = 2$ et $M = 0\ 125$

Les contenus de 8003 et 8002 à l'entrée seront les suivants : 8003 00 0000 0125

8002 00 0006 0008

Les mémoires 0 125 à 0 172 seront réservées aux données (matrice et seconds membres).

La 3ème ligne du système sera enregistrée dans les mémoires 0 141 à 0 148.

Les mémoires 0 173 à 0 184 seront les mémoires auxiliaires pour les calculs et résultats partiels.

Les résultats occuperont pour :

- la 1° série de seconds membres, les mémoires :

0 125, 0 133, 0 141, 0 149, 0 157, 0 165

- la 2° série de seconds membres les mémoires :

0 126, 0 134, 0, 142, 0 150, 0 158, 0 166

Forme du sous-programme : L'utilisateur dispose de deux paquets de cartes dont les contenus devront être enregistrés sur le tambour.

1°/ *l'A V F* (arithmétique en Virgule Flottante simple précision).

Ce programme occupe les mémoires 1 800 à 1 999 sauf les zones entrée et sortie standard de la bande 1 950 (1950 - 1960 et 1977 - 1986).

Il se présente sous forme de cartes «chargement 7 mots» (cartes roses), et est précédé d'un programme de chargement (cartes jaunes) qui utilise la bande 0000. Il faudra donc prévoir son chargement en temps utile dans le cas où l'on utiliserait la bande 0000 pour stocker les données ou à toute autre fin.

2°/ *Le SOUS-PROGRAMME* de résolution de systèmes linéaires en simple précision.

Il se présente sous forme numérique translatable «1 mot par carte» (cartes blanches).

Le facteur de translation Δ qui peut être quelconque mais inférieur ou égal à 1 500, sera chargé en 1 950.

La première instruction de ce sous-programme se trouve en Δ + 0000.

Encombrement : 200 mémoires pour l' A V F (obligatoirement de 1 800 à 1 999 , les zones entrée et sortie standard de la bande 1 950 étant disponibles).

300 mémoires pour le sous-programme.

Arrêt programme : Lorsque la matrice du système est singulière, nous avons l'arrêt :

01 0000 9999

Méthode utilisée : La méthode utilisée est la méthode de JORDAN avec choix du pivot maximal dans la matrice.

I, 1-1-2 SOUS-PROGRAMME en DOUBLE PRECISION.

Objet : Etant donné, en simple ou en double précision, la matrice et les vecteurs seconds membres d'un système linéaire, ce sous-programme calcule en virgule flottante *Flair* et en *Double Précision*, les vecteurs solutions.

Conditions d'utilisation : Si «*n*» désigne l'ordre de la matrice du système linéaire et «p» le nombre de vecteurs seconds membres, la limite d'utilisation de ce sous-programme est :

$$2(n+1)(n+p) + n \leqslant 1355$$

Ainsi avec p = 1, nous avons : $n_{MAX} = 24$

La virgule flottante simulée étant du type *Flair*, on représentera :

1 par $\underbrace{51\ 1000\ 0000}_{\text{I}}$ $\underbrace{00\ 0000\ 0000}_{\text{II}}$

Dans tout ce qui suit, nous appellerons I la partie gauche et II la partie droite d'un nombre en virgule flottante double précision.

Il faudra enregistrer l'A D P (arithmétique en Virgule Flottante Double Précision) sur le tambour.

Utilisation : Le système à résoudre doit être rangé ligne par ligne dans des mémoires consécutives du tambour. Chaque ligne se composant des *n* coefficients du premier nombre et des p coefficients des seconds nombres, c'est-à-dire d'une équation complète du système. Le rangement du système en double précision se fera de la façon suivante :

Un premier bloc de $(n + p)$ *n* mémoires dont la première a pour adresse M contiendra toutes les parties gauches des nombres en double précision.

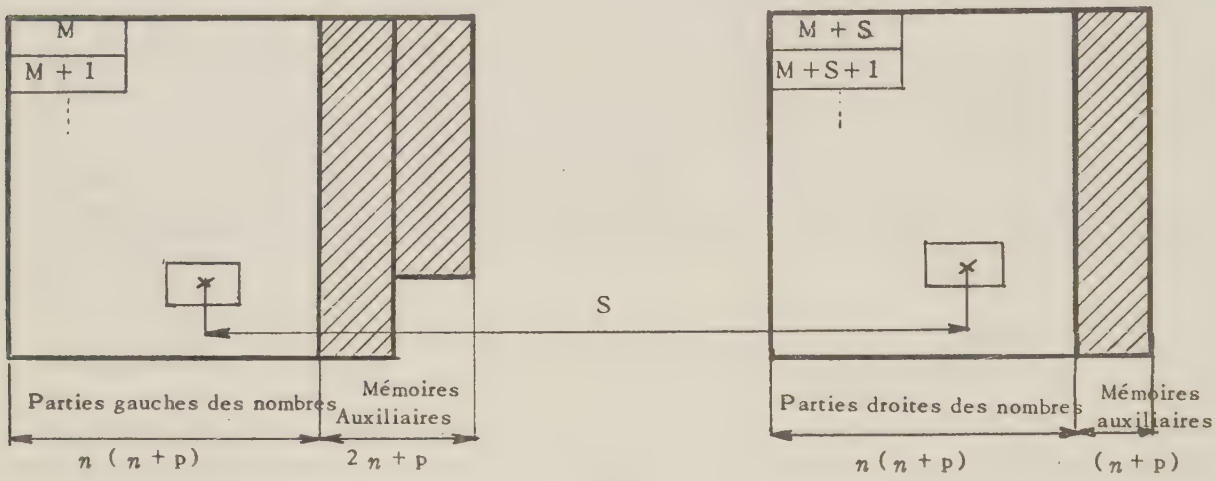
Un second bloc de $(n + p)$ *n* mémoires dont la première a pour adresse M + S (S > O) contiendra toutes les parties droites des nombres en double précision.

Le calcul utilise $[\,2(n + p) + n\,]$ mémoires auxiliaires pour les calculs et résultats intermédiaires. Ces mémoires auxiliaires se répartissent en deux blocs disposés de la façon suivante :

$(2n + p)$ mémoires en séquence, situées immédiatement à la suite du bloc de mémoires débutant en M et contenant les parties gauches des nombres en double précision,

$(n + p)$ mémoires en séquence, situées immédiatement à la suite du bloc de mémoires débutant en M + S et contenant les parties droites des nombres en double précision.

Nous aurons ainsi le schéma ci-dessous :



Remarque : Nous noterons qu'il faut $S \geq n\ (n + p) + 2n + p$

D'autre part, dans le cas où les données sont en simple précision, il ne faut pas oublier de s'assurer que les mémoires devant contenir, au cours des calculs, les parties droites des nombres en double précision sont bien initialement soit à zéro + soit à zéro -.

Avec les notations déjà définies, pour entrer dans le sous-programme de résolution des systèmes linéaires en double précision, l'utilisateur devra placer en :

8003	0	0		S		M
8002	0	0	0	0	n	$n + p$
8001		une instruction de sortie.				

Après le calcul, les mémoires qui contenaient les p premières colonnes de la matrice donnée, contiennent les p vecteurs solutions. Nous pouvons très bien avoir $p > n$. La seule limite étant celle due à la formule du paragraphe «Conditions d'utilisation».

Ainsi, par exemple, la $i^{è}$ racine de la j^{o} série de seconds membres (ou la $i^{è}$ composante du J^{o} vecteur solution) se trouvera dans les mémoires d'adresses :

$[M + j - 1 + (n + p)\ (i - 1)]$ pour la partie gauche

et

$[M + j - 1 + (n + p)\ (i - 1) + S]$ pour la partie droite.

Forme du Sous-Programme : L'utilisateur dispose de deux paquets de cartes dont les contenus devront être enregistrés sur le tambour.

1°/ *l'A D P* (Arithmétique en virgule flottante DOUBLE PRECISION).

Ce programme occupe les mémoires 1700 à 1999, sauf les zones entrée et sortie standard de la bande 1950. (1950 - 1960 et 1977 - 1986).

Il se présente sous forme de cartes «chargement 7 mots par carte» (cartes roses) et est précédé d'un programme de chargement (cartes jaunes) qui utilise la bande 0000. Il faudra donc prévoir son chargement en temps utile dans le cas où l'on désire utiliser la bande 0000 : soit pour stocker les données, soit à toute autre fin.

2°/ *Le SOUS-PROGRAMME* de résolution de systèmes linéaires Double précision.

Il se présente sous forme numérique translatable «1 mot par carte» (cartes blanches).

Le facteur de translation Δ qui peut être quelconque mais inférieur ou égal à 1355, sera chargé en 1950.

La première instruction de ce sous-programme se trouve en $\Delta + 0000$.

Encombrement : 300 mémoires pour l'A D P (obligatoirement 1700 à 1999 ; les zones entrée et sortie standard de la bande 1950 étant disponibles).

345 mémoires pour le sous-programme.

Arrêt programme : Lorsque la matrice du système est singulière, nous avons l'arrêt :

01 0000 9999

Méthode utilisée : La méthode utilisée est la méthode de JORDAN avec choix du pivot maximal dans toute la matrice.

I, 1-2 NOTICE EXPLICATIVE du SOUS-PROGRAMME en SIMPLE PRECISION

(voir page suivante)

Nous appellerons " n " l'ordre du système à résoudre, et " p " le nombre de vecteurs seconds membres.

Les données sont enregistrées sur le tambour à partir d'une mémoire d'adresse " M ". Le système est rangé équation par équation. Chaque équation comprend les premiers et les seconds membres.

Pour entrer dans le Sous Programme, il faut charger 8001, 8002, 8003 de la façon suivante:

(8001) = une instruction de sortie.
(8002) = oo (n) (n+p)
(8003) = oo oooo (M)

La première instruction du Sous Programme se trouve en 0000.

MEMOIRES de TRAVAIL

1°) à contenus constants

(0022) = 65 $\left[M + n(n + p)\right]$ 0105
Instruction utilisée pour l'appel de l'ordre initial "l_i" de l'inconnue à éliminer.

(0032) = 24 $\left[M + n(n + p)\right]$ 0204

(0040) = 69 $\left[M + n(n + p) + 1\right]$ 8002
Ces deux dernières instructions permettront de bloquer, à partir de la mémoire d'adresse $\left[M + n(n + p)\right]$, les $(n - k)$ mémoires associées aux inconnues restant à éliminer après la k° élimination.

(0052) = 61 $\left[M + n - 1 + n(n + p)\right]$ 0100
en position facteur, nous avons: la première adresse, diminuée de l'unité, de la série de "n + p" mémoires consécutives qui contiendront, pendant chaque élimination, l'équation pivot préalablement divisée par le pivot.

(0066) = 21 $\left[M + n + n(n + p)\right]$ 0254.
Instruction pour le stockage des termes de l'équation pivot, divisés par le pivot, dans une série de mémoires de travail consécutives.

(0115) = oo (n) oooo

(0116) = oo (p) oooo

(0125) = oo (n + p) oooo

(0134) = oo $\left[M + n - 1 + n(n + p) + p\right]$ oooo
Constante utilisée pour tester la fin du reclassemént de l'équation pivot, au cours de chaque élimination.

(0150) = 69 (M) 0153
Instruction pour l'appel des termes des "n" équations au cours de chaque élimination.

2°) à contenus variables.

(0001) = oo $(n - k + l_i - l_p + 1)$ oooo
Cette quantité représente, au cours de la k° élimination ($1 \leqslant k \leqslant n$), le rang qui devra être attribué à l'équation pivot, provisoirement stockée dans des mémoires de travail, dans le but d'obtenir des résultats convenablement rangés.
D'autre part:
- "l_i" représente l'ordre initial de l'inconnue éliminée au cours de la k° élimination.
  Nous avons: $1 \leqslant l_i \leqslant n$
- "l_p" représente l'ordre actuel de la même inconnue. Nous avons: $1 \leqslant l_p \leqslant n - k + 1$

(0003) = valeur exacte du pivot de l'élimination en cours.

(0009) = Mot d'ORIENTATION
Son contenu sera:
44 0107 0146 pour le choix du premier pivot.
44 0107 0158 dans tous les autres cas.

(0014) = oo $\overline{\text{oooo}}$ (n + p)

au tout début des calculs.

= 21 xxxx 0254

pour le stockage des termes de l'équation pivot divisés par le pivot, dans une série de mémoires de travail consécutives à partir de l'adresse $\left[M + n + n(n + p)\right]$.

Initialement, nous avons: (0014) = (0066)

= 21 xxxx 0157

pour le stockage des résultats de chaque élimination.

Au début de chaque élimination, nous avons:

xxxx = M - 1

(0016) = oo (l_p - 1) oooo

"l_p" représente l'ordre actuel de l'inconnue éliminée au cours de la k° élimination.

Remarque

Dans le cas du premier choix de pivot, nous avons:

(0016) = oo (n) oooo

(0021) = Mot d'ORIENTATION

Son contenu sera:

20 0109 0192 pour le choix du premier pivot.

20 0109 8001 dans tous les autres cas.

(0026) = 60 AdrP 0050

pour la division de l'équation pivot par le pivot.

Nous avons en adresse facteur: "AdrP" qui représente l'adresse de la mémoire contenant le pivot.

Le contenu de la mémoire 0026 sera utilisé pour éviter la division du pivot par lui-même.

= 69 xxxx 0153

pour les calculs de la k° élimination.

Ce contenu servira à tester dans chaque équation:

- le terme de même rang que le pivot.
  Nous aurons: "xxxx" = adresse de la mémoire contenant le terme de même rang que le pivot dans l'équation en cours de traitement.
- le dernier terme du premier membre.
  Nous aurons: "xxxx" = adresse, augmentée de l'unité, de la mémoire contenant le dernier terme de l'équation en cours de traitement.
  Ce terme est de rang (n - k + 1).
- le dernier second membre.
  Nous aurons: "xxxx" = adresse, augmentée de l'unité, de la mémoire contenant le dernier second membre de l'équation en cours de traitement.

(0028) = oo (k_p - 1) oooo

Au début de la k° élimination, "k_p" représente le rang actuel de l'équation pivot pour cette élimination. Ensuite, "k_p" représente le rang actuel de l'équation pivot pour l'élimination suivante.

Remarque

Au début du premier choix de pivot; nous avons:

(0028) = oo (n) oooo

(0033) = Mot d'ORIENTATION

Son contenu sera:

69 0203 0106 pour le choix du premier pivot.

et successivement:

65 0007 0265 et

60 0058 0273 dans tous les autres cas.

(0034) = oo xxxx oooo

Ce contenu servira à tester au cours de chaque élimination:

- l'équation pivot.
  Nous aurons: (0034) = (0028)
- l'équation de rang (n - k + 2). Il s'agit de détecter la fin du choix de pivot pour la prochaine élimination.
  Nous avons dans ce cas: (0034) = (0058)
- l'équation de rang (n - k + l_i - l_p + 2). Il s'agit de reclasser convenablement l'équation pivot de l'élimination en cours.
  Nous aurons: (0034) = (0001)
- la fin des calculs.
  Nous aurons dans ce dernier cas: (0034) = (0115)

(0039) = Au début de chaque élimination: la valeur absolue du pivot.
Ensuite, cette mémoire est remise à zéro.
Au cours de chaque élimination, cette mémoire contient la valeur absolue du pivot pour la prochaine élimi - nation.

(0045) = comptage des équations au cours de chaque élimination.
Ce compteur est remis à zéro au début de chaque élimination.

(0046) = Mot d'ORIENTATION associé aux équations.
Son contenu sera successivement au cours de chaque élimination:
oo oo88 oooo
oo oo89 oooo
oo oo9o oooo

(0058) = oo (n - k + 1) oooo
avec $1 \leqslant k \leqslant n$

(0059) = oo (l_p - 1) oooo
Au début de la k° élimination, "l_p" représente le rang actuel de l'inconnue à éliminer.
Ensuite, "l_p" représente le rang actuel de l'inconnue à éliminer au cours de l'élimination suivante.

(0064) = 60 xxxx 0050
pour l'appel des termes de l'équation pivot afin de les diviser par le pivot.
Initialement; nous avons: "xxxx" = l'adresse de la mémoire contenant le premier terme de l'équation pivot.

= 69 xxxx 0153
pour l'appel des termes de toutes les équations au cours de chaque élimination.
Nous avons initialement, xxxx = M .

(0072) = oo oooo oooo pour toutes les équations de rang inférieur à k_p (équation pivot) ou supérieur à ($n - k + l_i - l_p + 1$).

= oo (n + p) oooo pour toutes les équations de rang compris entre k_p et ($n - k + l_i - l_p + 1$).

(0078) = Initialisation de [illegible] du Sous Programme de Résolution de Systèmes Linéaires.

(0096) = Mot d'ORIENTATION associé aux termes d'une même équation.
Son contenu sera successivement au cours du traitement de chaque équation.
00 0088 0000
00 0089 0000
00 0090 0000

(0103) = Multiplicateur relatif à l'équation en cours de traitement.

(0109) = 61 xxxx 0100
Instruction d'appel des coefficients de l'équation pivot au cours de chaque élimination et pour le traitement de chaque équation. Il s'agit de l'équation pivot divisée par le pivot et stockée dans une série de mémoires de travail consécutives.
Au début du traitement de chaque équation, nous avons:
(0109) = (0052)

(0250) = 21 AdrP 0157
Au début de chaque élimination, "AdrP" représente l'adresse de la mémoire contenant la valeur exacte du pivot pour l'élimination en cours.
Ensuite, "AdrP" représente l'adresse de la mémoire contenant la valeur exacte du pivot pour l'élimination suivante.

INSTRUCTIONS

Initialisation.

```
0000   24 0078 0182
0182   69 8003 0038
0038   23 0014 0174      (0014) = oo oooo (n+p)
0174   69 0058 0224      (0058) = oo xxxx oooo
0224   22 0115 0274      (0115) = oo ( n ) oooo
0274   24 0028 0232
0232   24 0016 0275
0275   24 0058 0176
0176   35 0004 0088
0088   22 0125 0128      (0125) = oo (n+p) oooo
0128   16 0115 0226
0226   22 0116 0276      (0116) = oo ( p ) oooo
0276   65 8003 0083
0083   21 0045 0148      (0045) = oo oooo oooo
0148   24 0039 0142
```

```
0142   69 0150 0177      (0150) = 69 xxxx 0153
0177   22 0150 0227      (0150) = 69 ( M ) 0153
0227   10 0014 0277      (0014) = oo oooo (n+p)
0277   19 0115 0135      (0115) = oo ( n ) oooo
                         (8003) = oo ( M ) oooo
                         (8002) = oo (n(n+p)) oooo
0135   15 8003 0094      (8002) = oo [M+n(n+p)] oooo
```

en position facteur, la première adresse de la série de " n " mémoires utilisées pour la mise en ordre des résultats.

```
0094   15 0147 0178      (0147) = 69 oool 8002
0178   20 0040 0144      (0040) = 69 [M+1+n(n+p)] 8002
0144   16 0197 0228      (0197) = 04 0001 7897
0228   20 0022 0278      (0022) = 65 [M+n(n+p)] 0105
0278   69 0032 0185      (0032) = 24 xxxx 0204
0185   22 0032 0235      (0032) = 24 [M+n(n+p)] 0204
```

Chargement des mémoires utilisées pour la mise en ordre des résultats après chaque élimination. Au début des calculs, la mémoire d'adresse [M+n(n+p)+j-1] est associée à la j° inconnue et sera chargée avec oo (j-1) oooo.

```
0235   65 8002 0194
0194   15 0247 8002      (0247) = - 43 9999 9975
(1) 8002   21 xxxx 0130
0130   10 0007 0180      (0007) = oo oool oooo
0180   15 8001 0240
0240   11 0115 0230      (0115) = oo ( n ) oooo
0230   44 0133 0234
```

Le chargement des "n " mémoires n'est pas terminé.

```
0133   10 8001 8002  (1)
```

Le chargement des " n " mémoires est terminé.

```
0234   15 0138 0244      (0138) = oo oooo o124
0244   20 0066 0233      (0066) = 21 (M+n+n(n+p)) 0954
```

en position facteur, la première adresse d'une série de (n+p) mémoires de travail utilisées au stockage de l'équation pivot au cours de chaque élimination.

```
0233    15 0186 0041        (0186) = 39 9998 9846
0041    20 0052 0236        (0052) = 61 [M+n-1 ; +n(n+p)] 0100
0236    15 0116 0188        (0116) = oo ( p ) oooo
0188    69 8003 0145
0145    22 0134 0238        (0134) = oo [M+n-1 ; +n(n+p)+p] oooo
```

Préparation de la recherche du premier pivot.

```
0238    69 0091 0195        (0091) = 44 0107 0146
0195    24 0009 0141
0141    69 0245 0198        (0245) = 20 0109 0192
0198    24 0021 0191
0191    69 0200 0241        (0200) = 69 0203 0106
0241    24 0033 0076  (2)
```

Maintenant la recherche du premier pivot peut s'effectuer. Cette recherche présente un aspect particulier. En effet, et celà dès la première élimination, nous ferons le choix du pivot pour la (k+1)° élimination en même temps que nous ferons les calculs de la k° élimination.
Nous allons par conséquent exposer la suite du programme dans le cas général en mettant en évidence les embranchements particuliers à la recherche du premier pivot.

K° ELIMINATION.

Au début de cette élimination, nous avons " n " équations formées chacune de (n+p-k+1) termes. Les termes d'une même équation sont enregistrés dans des mémoires consécutives. Les premiers termes des " n " équations sont, et resterons toujours, dans les mémoires d'adresses: M ; M+(n+p); M+2(n+p); ..... ; M+(n-1)(n+p).

Le pivot qui a été choisi en même temps que s'effectuaient les calculs de l'élimination précédente (mis à part le cas du premier pivot) est caractérisé par:

- sa valeur absolue: contenu de la mémoire 0039
- son adresse: en position facteur du contenu de la mémoire d'adresse 0250
- " k_p ": le numéro de l'équation dans laquelle il a été trouvé.

  (0028) = oo (k_p-1) oooo
- " l_p ": le numéro de l'inconnue dont il est le coefficient.

  (0059) = oo (l_p-1) oooo

Notons que nous aurons toujours k_p et l_p inférieurs ou égaux à (n-k+1).

A ce moment, nous avons (n-k+1) mémoires associées aux inconnues restant à éliminer. Le contenu de l'une de ces mémoires va nous permettre de déterminer l'ordre initial de l'inconnue à éliminer à la k° étape du calcul. Connaissant cet ordre initial, nous pourrons décider de la place qu'occupera l'équation pivot à la fin de la k° élimination.

En effet, le contenu de la mémoire d'adresse: M+n(n+p)+l_p-1 nous donne l'ordre initial de la l_p° inconnue, que nous désignerons par l_1 .

$(M+n(n+p)+l_p-l) = oo\ (\ l_i-l\ )\ oooo$

La quantité l_i-l_p (toujours positive) représente le nombre d'inconnues précédant la l_i° qui ont déjà été éliminées. Les équations pivots correspondantes devront précéder l'équation pivot à classer. Par conséquent, cette équation sera précédée au total de:

$(n-k+l_i-l_p)$ équations.

Son premier coefficient sera à enregistrer dans la mémoire d'adresse:

$M+(n-k+l_i-l_p)(n+1)$

Finalement, avant de commencer les calculs de la k° élimination, nous bloquerons dans (n-k) mémoires consécutives les renseignement relatifs aux inconnues qui resteront à éliminer après la k° élimination. A ce moment la situation sera absolument identique à ce qu'elle était au début de la k° élimination, sauf que chaque équation ne sera constituée que de (n+p-k) termes.

En procédant de cette manière, à la fin des n éliminations, nous aurons d'une façon générale: la i° équation qui aura servie d'équation pivot lors de l'élimination de la i° inconnue.

Chaque équation sera constituée de p termes (n+p-k avec k=n) qui seront les p racines relatives aux p différents sedonds membres. On trouvera donc le vecteur solution relatif au i° vecteur second membre dans les mémoires d'adresses:

M+i-1
M+(n+p)+i-1
.....
M+(n-1)(n+p)+i-1

Autrement dit, les p vecteurs solutions occuperont les mémoires initialement réservées aux p premières colonnes de la matrice du système.

Restauration des aiguillages.

(3)	0200	69 0203 0106	(0203) = 44 0107 0158
(12)	0106	24 0009 0012	
	0012	69 0015 0018	(0015) = 20 0109 8001
	0018	24 0021 0024	
	0024	69 0027 0030	(0027) = 65 0007 0265
	0030	24 0033 0036	
	0036	60 0039 0043	(0039) = valeur absolue du pivot.
	0043	44 0047 0048	Le pivot est-il nul?

Le pivot est nul.

0048	01 0000 9999	La matrice du système est singulière, on ne peut pas poursuivre les calculs

Le pivot n'est pas nul.

0047	65 0250 0055	(0250) = 21 AdrP 0157 AdrP: adresse du pivot.
0055	15 0008 8002	(0008) = 48 oooo 0096
8002	69 AdrP 0253	
0253	24 0003 0156	(0003) = valeur exacte du pivot.
0156	65 0059 0013	(0059) = oo (l_p-l) oooo
0013	24 0016 0019	
0019	15 0022 8002	(0022) = 65 $\begin{bmatrix} M \\ +n(n+p) \end{bmatrix}$ 0105
8002	65 yyyy 0105	yyyy = $M+n(n+p)+l_p-l$ (8002) = oo (l_i-l) oooo
0105	15 0058 0063	(0058) = oo $(n-k+1)$ oooo
0063	16 0059 0113	(0059) = oo (l_p-l) oooo
0113	21 0039 0042	(0039) = oo oooo oooo Remise à zéro de la mémoire contenant la valeur maximum du pivot, en vue du prochain choix de pivot.
0042	24 0045 0098	
0098	22 0001 0154	(0001) = oo $(n-k+l_i-l_p+1)$ oooo

Blocage, à partir de la mémoire d'adresse M+n(n+p), des (n-k) mémoires associées aux inconnues restant à éliminer.après la k° élimination.

Au cours de cette élimination, l'inconnue d'ordre l_p disparait. Par conséquent, tous les coefficients des inconnues d'ordres supérieurs à l_p, qui restent à éliminer, vont gagner une position. Il faut donc que les mémoires associées à ces inconnues subissent un déplacement analogue pour que leurs contenus conservent toute leur signification.

En conséquence, nous commencerons à travailler sur la mémoire associée à la $(l_p+1)^e$ inconnue. Sonadresse est égale à $M+n(n+p)+l_p$.

```
      0154   60 0016 0071      (0016) = oo (l_p-1) oooo
      0071   15 8001 0029
      0029   15 0032 0037      (0032) = 24 [M+n(n+p)]  0204
(4)   0037   10 0040 8003      (0040) = 69 [M+1+n(n+p)]  8002
                               (8003) = 69 [M+l_p+n(n+p)]  8002
                               (8002) = 24 [M+l_p-1+n(n+p)]  0204
      8003   69 xxxx 8002
      8002   24 xxxx 0204
      0204   10 0007 0161      (0007) = oo oool oooo
      0161   10 8001 0069
      0069   15 8001 0077
      0077   11 0040 0095      (0040)
      0095   11 0058 0163      (0058) = oo (n-k+1) oooo
      0163   46 0017 0068      Test de fin de blocage des (n-k)
                               mémoires associées aux inconnues
                               restant à éliminer.
```

Le blocage n'est pas terminé.

```
      0017   10 8001 0025
      0025   11 0007 0037  (4)
```

Transfert de l'équation pivot dans des mémoires de travail.

Nous allons diviser les (n+p-k+1) termes de l'équation pivot, situés dans des mémoires consécutives à partir de l'adresse $M+(k_p-1)(n+p)$, par le pivot d'adresse $AdrP = M+(k_p-1)(n+p)+l_p-1$.

La division du pivot par lui-même donne l'unité. Nous n'effectuerons pas ce calcul, et nous ne mettrons pas le résultat correspondant en mémoire.

Les (n+p-k) autres résultats seront mis dans des mémoires consécutives de travail à partir de l'adresse M+n(n+p)+n.

```
      0068   60 0250 0155      (0250) = 21 AdrP 0157
      0155   10 0108 0213      (0108) = 38 9999 9893
      0213   15 0066 0121      (0066) = 21 [M+n+n(n+p)]  0057
      0121   21 0026 0079      (0026) = 60 AdrP 0050
      0079   11 0016 0171      (0016) = oo (l_p-1) oooo
(5)   0171   21 0064 0067      (0064) = 60 (AdrP-l_p+1) 0050
                               AdrP-l_p+1 = M+(k_p-1)(n+p)
      0067   20 0014 0117      (0014) = 21 [M+n+n(n+p)]  0057
      0117   11 0026 0031      (0026)
      0031   44 0035 0086
```

Le prochain terme à diviser est le pivot. Nous passons à la division suivante sans effectuer d'opération.

```
      0086   60 0064 0219      (0064)
      0219   15 0014 0269      (0014)
      0269   10 0007 0171 (5)  (0007) = oo oool oooo
```

Le prochain terme à diviser n'est pas le pivot, nous effectuons l'opération.

```
      0035   10 0016 0221      (0016) = oo (l_p-1) oooo
      0221   11 0058 0263      (0058) = oo (n-k+1) oooo
      0263   11 0116 0271      (0116) = oo ( p ) oooo
```

0271 44 0064 0076 Test de fin des divisions.

Toutes les divisions n'ont pas été effectuées.

0064 60 xxxx 0050

Entrée dans l'A.V.F. pour effectuer une division.

0050	69 0003 0006	(0003) = diviseur.
0006	24 1801 0004	
0004	69 1900 0053	VDV
0053	24 1808 0011	
0011	69 0014 1800	(0014) = instruction de sortie.

1800	
Sous-Programme A. V. F.	
1803	(1803) = (0014) = 21 xxxx 0254

Mise en mémoire du résultat.

1803 21 xxxx 0254

Pré paration de la division suivante.

0254	60 0064 0119	(0064)
0119	15 0014 0169	(0014)
0169	15 0007 0211	(0007) = oo oool oooo
0211	10 8001 0171 (5)	

Toutes les divisions sont terminées, nous commençons les calculs d'élimination.
Pour ces calculs, nous devrons répondre aux impératifs suivants:

1°) ne faire aucun calcul avec l'équation pivot.
Il s'agit de la k_p° équation.

2°) limiter le choix de pivot pour l'élimination suivante aux seules équations n'en ayant pas encore fourni.
Ceci s'adresse aux (n-k+1) premières équations sauf la k_p°, qui est l'équation pivot de l'élimination en cours.

3°) reclasser l'équation pivot de l'élimination en cours.
Elle sera à reclasser derrière les (n-k) équations précédentes, en position de $(n-k+1-l_p+1)$° équation.

4°) détecter la fin de l'élimination.

Ces quatre impératifs feront l'objet d'un test situé dans la mémoire 0089

(0089) = 44 0143 0033

Les trois premiers dépendrons, après le test, d'un aiguillage du type 88, 89, 90 (contenus successifs de la mémoire 0046)
Le quatième sera exécuté par la modification du contenu de la mémoire 0033

5°) à l'intérieur de chaque équation, repérer le terme de même rang que le pivot.
Il s'agit du l_p° terme.

6°) à l'intérieur de chaque équation, tester la fin du premier membre ((n-k+1) termes)

7°) à l'intérieur de chaque équation, détecter la fin des seconds membres; c'est à dire de l'équation.

Ces trois derniers impératifs feront l'objet d'un test situé dans la mémoire de travail 0009

(0009) = 44 0107 0158

et l'aiguillage sera du type 88, 89, 90 (contenus successifs de la mémoire 0096)

Signalons aussi les contenus de diverses mémoires de travail au début de ces calculs.

1°) <u>Dans le cas général</u>.

(0009) = 44 0107 0158
(0021) = 20 0109 8001
(0033) = 65 0007 0265
(0039) = 00 0000 0000

(0045) = 00 0000 0000

2°) Dans le cas du choix du premier pivot.

(0009) = 44 0107 0146
(0021) = 20 0109 0192
(0016) = 00 (n) 0000
(0028) = 00 (n) 0000
(0033) = 69 0203 0106
(0039) = 00 0000 0000
(0045) = 00 0000 0000

Enfin, sous forme de " REMARQUES " nous indiquerons les embranchements particuliers à la recherche du premier pivot.

(2) 0076 65 0150 0205 (0150) = 69 (M) 0153

REMARQUE C'est par cette instruction que commence le choix du premier pivot.

0205 24 0064 0167 (0064)
0167 21 0072 0075 (0072) = oo oooo oooo
0075 69 0028 0081 (0028) = oo (k_p-1) oooo
dans le cas général.
oo (n) oooo
dans le cas du choix du premier pivot.
0081 24 0034 0087 (0034)
0087 69 0090 0093 (0090) = oo oo88 oooo
0093 24 0046 0099 (0046)
0099 15 0002 0057 (0002) = - 48 oooo 9996
(8002) = 21 (M) 0157
(6) 0057 20 0014 0217 (0014)
(7) 0217 11 0034 0089

0089 44 0143 0033 Test relatif aux équations.

L'équation que nous devons traiter, n'est:
- ni l'équation pivot
- ni la (n-k+2)° équation
- ni la (n-k+l_2-l_p+2)° équation.

Les n équations du système n'ont pas été toutes traitées.

0143 65 0064 0020
0020 15 0016 0122 (0016) = oo (l_p-1) oooo
dans le cas général.
oo (n) oooo
dans le cas du choix du premier pivot.
0122 20 0026 0129 (0026) = 69 xxxx 0153
0129 15 0082 8002 (0082) = oo oooo olo2
8002 69 yyyy 0255 yyyy = adresse du terme de même rang que le pivot dans l'équation traitée.
0255 24 0103 0206 (0103) = multiplicateur pour les calculs d'élimination relatifs à une équation.
0206 69 0090 0193 (0090) = oo oo88 oooo
0193 24 0096 0149 (0096)
0149 69 0052 0256 (0052) = 61 $\left[\begin{matrix}M+n-1\\ +n(n+p)\end{matrix}\right]$ 0100
0256 24 0109 0062 (0109)
(8) 0062 60 0064 0070
0070 11 0026 0009

0009 Mémoire de travail utilisée pour l'orientation du programme.

Pour l'étude du choix du premier pivot, se reporter à la REMARQUE N°1.

Nous poursuivons l'étude du programme dans le cas général.

0009 44 0107 0158 Test relatif aux termes d'une même équation.

Le terme sur lequel nous devons opérer n'est pas:

- de même rang que le pivot.
- de rang (n-k+2). Fin du traitement du premier membre.
- le dernier terme de l'équation complète.

```
0107    60 0014 0120        (0014)
0120    15 0109 0114        (0109)
0114    10 0007 0261        (0007) = oo oool oooo
0261    15 8001 0170
0170    21 0014 0021
```

0021 Mémoire de travail utilisée pour l'orientation du programme.

Pour l'étude du choix du premier pivot, se reporter à la REMARQUE N°2.

Nous poursuivons l'étude du programme dans le cas général.

```
0021    20 0109 8001
8001    61 xxxx 0100
```

Entrée dans l'A.V.F. pour éxécuter une multiplication.

```
0100    69 0103 0056        (0103) = multiplicateur
0056    24 1801 0054
0054    69 1996 0049        VMU
0049    24 1808 0061
0061    69 0064 1800        (0064) = instruction de sortie
```

```
1800
Sous-Programme
A. V. F.
1803
```

(1803) = (0064)
= 69 xxxx 0153

Séquence d'Entrée dans l'A.V.F. pour effectuer une addition.

```
1803    69 xxxx 0153
0153    24 1801 0104
0104    69 1902 0005        VAA
0005    24 1808 0111
0111    69 0014 1800        (0014) = instruction de sortie
```

```
1800
Sous-Programme
A. V. F.
1803
```

(1803) = (0014)
= 21 xxxx 0157

Mise en mémoire du résultat.

```
1803    21 xxxx 0157
0157    69 0046 0199        (0046) = oo oo8/9x oooo
0199    96 0102 0207
```

L'équation en cours de traitement fait partie des (n-k+1) premières équations qui doivent participer au choix de pivot pour la prochaine élimination.

```
0102    69 0096 0249        (0096) = oo oo8/9x oooo
0249    96 0152 0207
```

Nous travaillons sur le premier membre d'une équation, le choix de pivot doit avoir lieu.

```
0152    67 8003 0159
0159    16 0039 0243        (0039) = valeur absolue du
                            pivot actuellement retenu.
0243    46 0207 0097
```

Le dernier résultat obtenu est le maximum en valeur absolue et doit être retenu comme pivot.

```
0097    15 8001 0257
0257    20 0039 0092        (0039) = nouvelle valeur absolue
                            du pivot actuellement retenu.
0092    65 0045 0051        (0045) = compteur d'équations.
0051    69 0046 0101        (0046) = oo oox8/9 oooo
0101    95 0208 0258
0258    16 0007 0208        (0007) = oo oool oooo
0208    20 0028 0131        (0028) = en position facteur;
                            le numéro d'ordre de l'équation
                            ayant fourni le pivot, moins un.
0131    65 0109 0164        (0109) = 61 xxxx 0100
0164    16 0052 0209        (0052) = 61 [M+n-1 / +n(n+p)] 0100
0209    16 0007 0112        (0007)
0112    20 0059 0162        (0059) = en position facteur
```

le rang moins un, du pivot retenu, dans son équation.

```
0162    69 0014 0267        (0014)
0267    24 0250 0207        (0250) = en position facteur, l'adresse de la mémoire qui contient la valeur exacte du terme retenu comme pivot.
```

Le choix du pivot est terminé.
Il n'y a pas eu de choix de pivot.
Dans les deux cas, préparation du calcul suivant.

```
(9)  0207    60 0064 0220        (0064)
     0220    10 0007 0212        (0007) = oo oool oooo
     0212    21 0064 0062  (8)
```

Réponse négative au test: 44 0107 0158.
La série de calculs en cours sur une partie d'équation est terminée.

```
0158    60 8001 0118        (8001) = 69 xxxx 0153
0118    15 0007 0262        (0007)
0262    15 0096 0151        (0096) = oo oo8/98/9 oooo
0151    96 0259 0010
0259    95 0214 0264
```

Tous les termes de l'équation, de rang inférieur à celui du pivot, ont subi les calculs d'élimination.
Préparation d'une nouvelle série de calculs qui portera sur tous les autres termes du premier membre.

```
0214    11 0016 0172        (0016) = oo (l_p-1) oooo
0172    10 0058 0065        (0058) = oo (n-k+1) oooo
0065    21 0026 0179
0179    20 0096 0207  (9)   (0096) = oo oo89 oooo
```

Remarque: En 0207, nous modifions l'instruction d'appel des coefficients de l'équation, de façon à ne faire aucun calcul avec le terme de même rang que le pivot.

Par contre, nous ne modifions pas l'instruction de mise en mémoire des résultats.

Tout le premier membre de l'équation a subi les calculs d'élimination.
Préparation, pour cette équation, de la fin des calculs qui porteront sur les "p" seconds membres.
Nous aurons un 9 en position 6 du contenu de 0096, d'où la suppression du choix du pivot.

```
0264    10 0116 0222        (0116) = oo ( p ) oooo
0222    21 0026 0229
0229    20 0096 0062  (8)   (0096) = oo oo90 oooo
```

L'équation entière a subi les calculs d'élimination.
Nous allons préparer le passage à l'équation suivante.

```
0010    10 0115 0270        (0115) = oo ( n ) oooo
0270    11 0058 0165        (0058) = oo (n-k+1) oooo
0165    21 0064 0168        (0064) = 69 xxxx 0153
                            l'adresse facteur est celle
                            du premier terme de l'équation
                            suivante.
0168    10 0002 0060        (0002) = - 48 oooo 9996
0060    11 0072 0127        (0072) = voir la remarque
                                     ci-dessous.
0127    21 0014 0218        (0014) = intruction pour la
                            mise en mémoire des résultats
                            suivants.
(10) 0218    60 0045 0201        (0045) = comptage des équationS.
     0201    10 0007 0215        (0007) = oo oool oooo
     0215    21 0045 0217  (7)
```

Remarque:

- (0072) = oo oooo oooo, pour toutes les équations de rang inférieur à k_p (équation pivot) ou supérieur à $(n-k+1_i-1_p+1)$.

La nouvelle équation obtenue par élimination sera mise à la place de l'équation dont elle est issue.

- (0072) = oo (n+p) oooo, pour toutes les équations de rang compris entre k_p et $(n-k+l_i-l_p+1)$.

La nouvelle équation obtenue par élimination sera mise à la place de l'équation précédant celle dont elle est issue.

Réponses négatives au test: 44 0143 0033

1°) Cas du choix du premier pivot.

Le choix du premier pivot est terminé, nous sommes prêts à commencer la première élimination.

0033 69 0203 0106 (12)

2°) Cas général.

Première, deuxième ou troisième réponse négative au test: 44 0143 0033.

L'équation rencontrée présente une particularité. Le programme doit être ajusté ou modifié.

0033	65 0007 0265		(0007) = oo oool oooo
0265	15 0046 0251		(0046) = oo oo8/98/9 oooo
0251	96 0110 0160		
0110	95 0166 0216		

L'équation que nous sommes prêts à traiter est la k_p° équation. Elle est " équation pivot " pour la k° élimination et ne doit pas subir de calcul. Nous passons à l'équation suivante après préparation du test sur la $(n-k+2)$° équation.

0166	20 0046 0202		(0046) = oo oo89 oooo
0202	69 0058 0266		(0058) = oo (n-k+1) oooo
0266	24 0034 0137		
0137	65 0064 0272		(0064) = 69 xxxx 0153
0272	15 0125 0279		(0125) = oo (n+p) oooo
0279	24 0072 0175		(0072) = oo (n+p) oooo
0175	20 0064 0218	(10)	

L'équation que nous sommes prêts à traiter est la $(n-k+2)$° équation. Le choix du pivot pour l'élimination suivante doit être supprimé pour les autres équations.(9 en position 6 du contenu de 0046)

Préparation du test sur la $(n-k+l_i-l_p+2)$° équation.

0216	20 0046 0252		(0046) = oo oo90 oooo
0252	69 0001 0210		(0001) = oo $(n-k+l_i-l_p+1)$ oooo
0210	24 0034 0187		
0187	60 0045 0217	(7)	(0045) = comptage des équations.

L'équation que nous sommes prêts à traiter est la $(n-k+l_i-l_p+2)$° équation. Avant d'effectuer ce travail, il faut ranger l'équation pivot en position de $(n-k+l_i-l_p+1)$° équation, de façon à obtenir des résultats parfaitement rangés après les "n" éliminations.

Préparation du test sur la n° équation et modification du contenu de la mémoire de travail 0033.

0160	69 0115 0268	(0115) = oo (n) oooo
0268	24 0034 0237	
0237	69 0140 0044	(0140) = 60 0058 0273
0044	24 0033 0136	(0033)

Nous allons translater les contenus des (n+p-k) mémoires de travail consécutives à partir de $\left[M+n(n+p)+n\right]$ dans des mémoires consécutives à partir de $\left[M+(n-k+l_i-l_p)(n+p)\right]$.

	0136	60 0052 0260	(0052) = 61 $\left[\begin{matrix}M+n-1\\+n(n+p)\end{matrix}\right]$ 0100
	0260	15 0014 0023	(0014) = 21 zzzz 0157
			zzzz = $M+(n-k+l_i-l_p)(n+p)-1$
	0023	10 0126 0181	(0126) = 08 oooo 7902
	0181	15 0084 0139	(0084) = 02 9999 9982
			(8002) = 24 zzzz 0139
			(8003) = 69 $\left[\begin{matrix}M+n-1\\+n(n+p)\end{matrix}\right]$ 8002
(11)	0139	10 0007 0073	(0007) = oo oool oooo
	0073	15 8001 0231	
	0231	11 0134 0189	(0134) = oo $\left[\begin{matrix}M+n-1\\+n(n+p)+p\end{matrix}\right]$ oooo
	0189	11 0058 0123	(0058) = oo (n-k+1) oooo
	0123	69 8003 0080	
	0080	99 0184 0085	

La translation n'est pas terminée.

```
0184    10 0134 0239          (0134)
0239    10 0058 8003          (0058)
8003    69 xxxx 8002
8002    24 xxxx 0139   (11)
```

La translation est terminée.
Préparation des calculs pour l'équation suivante.

```
0085    16 0084 0190          (0084) = 02 9999 9982
0190    16 0058 0173          (0058) = oo (n-k+1) oooo
0173    15 0115 0223          (0115) = oo ( n ) oooo
0223    11 8003 0132
0132    21 0072 0225          (0072) = oo oooo oooo
0225    10 0045 0057   (6)    (0045) = comptage des équations.
```

Quatrième réponse négative au test: 44 0141 0033
La k° élimination est terminée.

```
0033    60 0058 0273          (0058)
0273    11 0007 0074          (0007) = oo oool oooo
0074    21 0058 0124
0124    44 0200 0078
```

Les "n" éliminations n'ont pas été effectuées.

```
0200    Nous commençons une nouvelle élimination.
        Voir en (3).
```

La résolution du système linéaire est terminée.

```
0078    Nous effectuons l'instruction placée en 8001
        lors de l'ENTREE dans le SOUS-PROGRAMME de
        RESOLUTION de SYSTEMES LINEAIRES.
```

REMARQUES concernant le choix du premier pivot.

REMARQUE N° 1

```
0009    44 0107 0146  Test de fin du premier membre
                      d'une équation.
```

L'examen du premier membre de l'équation n'est pas terminé.

```
0107    nous reprenons le cours du programme général.
        (page 97)
```

L'examen du premier membre de l'équation est terminé.
Nous préparons le passage à l'équation suivante en supprimant l'étude des seconds membres.

```
0146    60 8001 0246          (8001) = 69 xxxx 0153
0246    10 0116 0165          (0116) = oo ( p ) oooo
                              (8003) = 69 xxxx 0153
                              en adresse facteur, nous avons
                              l'adresse de la mémoire
                              contenant le premier terme de
                              l'équation suivante.
0165    nous reprenons le cours du programme général.
        (page 98)
```

REMARQUE N° 2

```
0021    20 0109 0192          (0109) = 61 xxxx 0100
0192    65 0064 0242          (0064) = 69 xxxx 0153
0242    15 0196 8002          (0196) = - 01 9999 9994
8002    67 xxxx 0159
0159    nous reprenons le cours normal du programme général.
        (page 97)
```

I, 2 - *SOUS-PROGRAMME de RESOLUTION de SYSTEMES LINEAIRES et d'INVERSION de MATRICE en SIMPLE PRECISION et en DOUBLE PRECISION. METHODE de JORDAN*

(Références : J S P 4 et J D P 4)

I, 2-1 - NOTICES d'UTILISATION

I, 2-1-1 SOUS-PROGRAMME en SIMPLE PRECISION

Objet : Etant donné la matrice et les vecteurs seconds membres d'un système linéaire, ce sous-programme calcule, en virgule flottante *Flair* et en simple précision, la matrice inverse et les vecteurs solutions.

Conditions d'Utilisation : si «*n*» désigne l'ordre de la matrice du système linéaire et «p» le nombre de vecteurs seconds membres, la limite d'utilisation de ce sous-programme est :

$$(n+1)\ (n+p) + n \leq 1\,470$$

Ainsi avec p = 1, nous avons : $n_{MAX} = 36$

La virgule flottante simulée étant du type *Flair*, on représentera :

1 par 51 1000 0000 +

Il faudra enregistrer l'A V F (Arithmétique en Virgule Flottante simple précision) sur le tambour

Utilisation : Le système à résoudre doit être rangé ligne par ligne dans des mémoires consécutives du tambour. Chaque ligne se compose des *n* coefficients du premier membre et des p coefficients des seconds membres, c'est-à-dire d'une équation complète du système.

Lecalcul utilise (2 *n* + p) mémoires auxiliaires pour les calculs et résultats intermédiaires. Ces (2 *n* + p) mémoires doivent se trouver en séquence, le bloc qu'elles occupent étant situé immédiatement après celui occupé par les données.

Si nous désignons par M l'adresse de la première mémoire du bloc où est rangé le système (ligne par ligne), l'utilisateur devra placer en :

8003 0 0 0 0 0 0 M +

8002 0 0 0 0 *n* *n* + p +

8001 une instruction de sortie.

Après le calcul, les mémoires qui contenaient les coefficients de la matrice donnée contiennent ceux de la matrice inverse.

Les mémoires qui contenaient les vecteurs seconds membres contiennent les vecteurs solutions correspondants. La valeur de p peut être quelconque. Elle n'est limitée que par la formule du paragraphe précédent.

Ainsi, par exemple, si nous avons :

$$n = 6,\quad p = 2 \quad \text{et} \quad M = 0125$$

les mémoires 0125 à 0172 contiennent les données, la 3° ligne du système par exemple occupe les mémoires 0141 à 0148, les mémoires 0173 à 0185 seront les mémoires auxiliaires pour les calculs et résultats partiels.

Une fois les calculs terminés :
les résultats occupent les mémoires 0125 à 0172, (la matrice inverse se trouve à la place de la matrice donnée); de cette façon, les mémoires 0141 à 0146 contiennent la 3° ligne de la matrice inverse. Les vecteurs solutions se trouvent à la place des vecteurs seconds membres, c'est-à-dire :
le 1° vecteur solution dans les mémoires 0131, 0139, 0147, 0155, 0163, 0171 ,
le 2° vecteur solution dans les mémoires 0132, 0140, 0148, 0156, 0164, 0172.

Forme du sous-programme : l'utilisateur dispose de deux paquets de cartes dont les contenus seront à enregistrer sur le tambour.

1° l'*A V F* (Arithmétique en Virgule Flottante simple précision).

Ce programme occupe obligatoirement les mémoires 1800 à 1999 sauf les zones entrée et sortie standard de la bande 1950 (1950 - 1960 et 1977 - 1986).

Il se présente sous forme de cartes : «chargement 7 mots par carte» (cartes roses) et est précédé d'un programme de chargement (cartes jaunes), qui utilise la bande 0000. Il faudra donc prévoir son chargement en temps utile dans le cas où l'on désire utiliser la bande 0000 pour stocker les données ou à toute autre fin.

2° *Le SOUS-PROGRAMME* de Résolution de Système Linéaire et d'inversion de matrice.

Il se présente sous forme numérique translatable «1 mot par carte». (cartes blanches).

Le facteur de translation Δ qui peut être quelconque mais inférieur ou égal à 1470, sera chargé en 1950.

La première instruction de ce sous-programme se trouve en $\Delta + 0000$.

Encombrement : 200 mémoires pour l'A V F (obligatoirement de 1800 à 1999, les zones entrée et sortie standard de la bande 1950 étant disponibles).

330 mémoires pour le sous-programme.

Arrêt programmé : Lorsque la matrice du système est singulière nous avons l'arrêt :

01 0000 9999

Méthode utilisée : La méthode utilisée est la méthode de JORDAN avec choix du pivot maximal dans toute la matrice.

I, 2-1-2 SOUS-PROGRAMME en DOUBLE PRECISION.

Objet : Etant donné la matrice et les vecteurs seconds membres d'un système linéaire, ce Sous-Programme calcule, en Virgule Flottante *Flair* et en *Double Précision*, la matrice inverse et les vecteurs solutions.

Conditions d'utilisation : si «n» désigne l'ordre de la matrice du système linéaire et «p» le nombre de vecteurs seconds membres, la limite d'utilisation de ce Sous-Programme est :

$$2\,(n+1)\,(n+p) + n \leq 1300$$

Ainsi, avec p = 1, nous avons $n_{MAX} = 24$.

La virgule flottante simulée étant du type *Flair*, on représentera :

1 par $\underbrace{51\quad 1000\quad 0000}_{\text{I}}\qquad \underbrace{00\quad 0000\quad 0000}_{\text{II}}$

Dans tout ce qui suit, nous appellerons I, la partie gauche et II, la partie droite d'un nombre en virgule flottante Double Précision.

Il faudra enregistrer l'A D P (Arithmétique en Virgule Flottante Double Précision) sur le tambour.

Utilisation : Le système à résoudre doit être rangé ligne par ligne dans des mémoires consécutives du tambour. Chaque ligne se compose des n coefficients du premier membre et des p coefficients des seconds membres ; c'est-à-dire d'une équation complète du système. Le rangement du système en double précision se fera de la façon suivante :

un premier bloc de $(n + p)\ n$ mémoires dont la première a pour adresse M contiendra toutes les parties gauches des nombres en double précision.

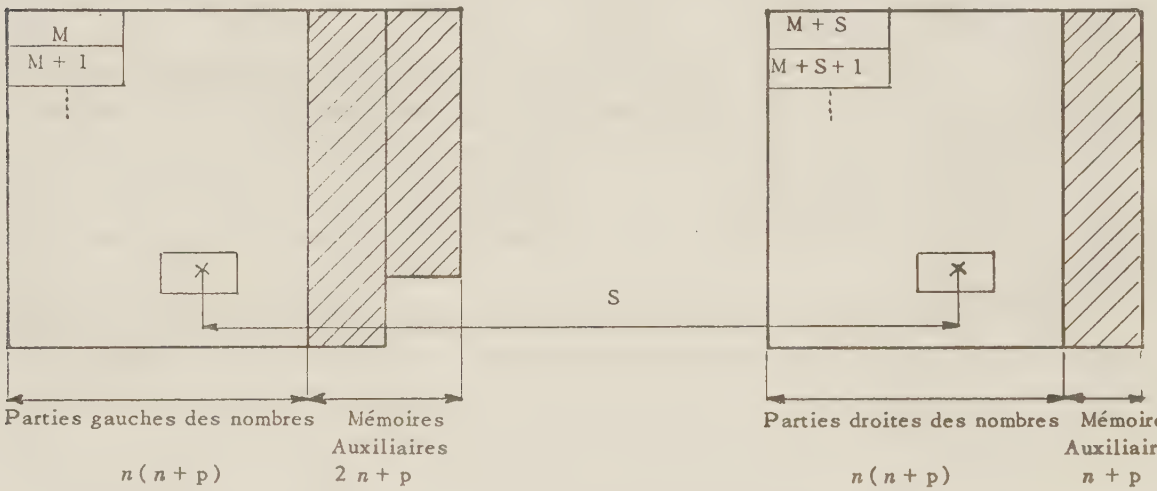
un second bloc de $(n + p)\ n$ mémoires dont la première a pour adresse $M + S\ (S > 0)$ contiendra toutes les parties droites des nombres en double précision.

Le calcul utilise $[\ 2(n + p) + n\ ]$ mémoires auxiliaires pour les calculs et résultats intermédiaires. Ces mémoires auxiliaires se répartissent en deux blocs disposés de la façon suivante :

$(2\ n + p)$ mémoires en séquence, situées immédiatement à la suite du bloc de mémoires débutant en M et contenant les parties gauches des nombres en double précision.

$(n + p)$ mémoires en séquence, situées immédiatement à la suite du bloc de mémoires débutant en $M + S$ et contenant les parties droites des nombres en double précision.

Nous aurons ainsi le schéma suivant :



Remarques : Nous noterons qu'il faut $S \geqslant n\ (n + p) + 2\ n + p$

D'autre part, dans le cas où les données sont en SIMPLE PRECISION, il ne faut pas oublier de s'assurer que les mémoires devant contenir, au cours des calculs, les parties droites des nombres en double précision sont bien initialement soit à zéro + , soit à zéro -.

Etant donné les notations déjà définies, pour entrer dans le Sous-Programme de résolution des systèmes linéaires et d'inversion de matrices en double précision, l'utilisateur devra placer en :

8003 0 0 S M

8002 0 0 0 0 n $n + p$

8001 une instruction de sortie.

Après le calcul, les mémoires qui contenaient les coefficients de la matrice donnée contiennent ceux de la matrice inverse.

Les mémoires qui contenaient les vecteurs seconds membres contiennent les vecteurs solutions correspondants. La valeur de p peut être quelconque. Elle n'est limitée que par la formule du paragraphe précédent.

Ainsi, par exemple, si nous avons :

$n = 6$, $p = 2$, M = 0125 et S = 0102 ,

les mémoires 0125 à 0172 contiennent les parties gauches des données ; les mémoires 0227 à 0274 contiennent les parties droites. Ainsi, la 3° ligne du système occupe les mémoires 0141 à 0148 et 0243 à 0250. Les mémoires 0173 à 0185 et 0275 à 0282 seront les mémoires auxiliaires pour les calculs et résultats intermédiaires. Une fois les calculs terminés : les résultats occupent les mémoires 0125 à 0172 et 0227 à 0274 ; la matrice inverse se trouve à la place de la matrice donnée, par exemple les mémoires 0141 à 0146 et 0243 à 0248 contiennent la troisième ligne de la matrice inverse.

Les vecteurs solutions se trouvent à la place des vecteurs seconds membres, c'est-à-dire :

- le 1° vecteur solution occupe les mémoires : 0131, 0139, 0147, 0155, 0163, 0171 et 0233, 0241, 0249, 0257, 0265, 0273.

- le 2° vecteur solution occupe les mémoires : 0132, 0140, 0148, 0156, 0164, 0172 et 0234, 0242, 0250, 0258, 0266, 0274 .

Forme du sous-programme : l'utilisateur dispose de deux paquets de cartes dont les contenus seront à enregistrer sur le tambour.

1°/ *l'A D P* (Arithmétique en virgule flottante DOUBLE PRECISION). Ce programme occupe les mémoires 1700 à 1999, sauf les zones entrée et sortie standard de la bande 1950 (1950 à 1960 et 1977 à 1986) :

Il se présente sous forme de cartes «chargement 7 mots par carte» (cartes roses) et est précédé d'un programme de chargement (cartes jaunes) qui utilise la bande 0000. Il faudra donc prévoir son chargement en temps utile dans le cas où l'on désire utiliser la bande 0000 : soit pour stocker les données ,soit à toute autre fin.

2°/ *Le SOUS-PROGRAMME* de Résolution de Systèmes Linéaires et d'inversion de Matrices en Double Précision. Il se présente sous forme numérique translatable « 1 mot par carte » (cartes blanches). Le facteur de translation Δ qui peut être quelconque mais inférieur ou égal à 1300 , sera chargé en 1950.

La première instruction de ce Sous-Programme se trouve en Δ + 0000 .

Encombrement : 300 mémoires pour l'A D P (obligatoirement 1700 à 1999 ; les zones entrée et sortie standard de la bande 1950 étant disponibles).

400 mémoires pour le sous-programme.

Arrêt programme : Lorsque la matrice du système est singulière, nous avons l'arrêt :

01 0000 9999

Méthode utilisée : La méthode utilisée est la méthode JORDAN avec choix du pivot maximal dans toute la matrice.

I, 2-2 NOTE EXPLICATIVE du SOUS-PROGRAMME en SIMPLE PRECISION.

(voir page suivante)

Nous appellerons "n" l'ordre du système linéaire à résoudre, et "p" le nombre de vecteurs seconds membres.

Les données sont enregistrée sur le tambour à partir d'une mémoire d'adresse "M". Le système est rangé équation par équation. Chaque équation comprend le premier et les seconds membres, soit au total (n + p) termes.

Pour entrer dans le Sous Programme, il faut charger 8001, 8002, 8003 de la façon suivante:

(8001) = une instruction de sortie.
(8002) = oo (n) (n + p)
(8003) = oo oooo (M)

La première instruction du Sous Programme se trouve en 0000.

MEMOIRES de TRAVAIL

1°) <u>à contenus constants</u>.

(0010) = oo $\boxed{M + n - 1 + (n + 1)(n + p)}$ oooo
Constante utilisée pour tester la fin du reclassement de l'équation pivot, au cours de chaque élimination. Nous avons en fait l'adresse de la dernière mémoire du bloc débutant en M et qui se compose de:
- n(n + p) mémoires pour les données.
- n mémoires pour le réarrangement des résultats.
- (n + p) mémoires pour le stockage de l'équation pivot au cours de chaque élimination.

(0022) = 55 $\boxed{M + n(n + p)}$ 0105
Instruction utilisée pour l'appel de l'ordre initial "k_i" de l'inconnue à éliminer.

(0044) = 65 $\boxed{M + n(n + p)}$ 0155
Instruction utilisée pour l'appel de l'ordre initial "l_i" de l'inconnue à éliminer.

(0045) = 15 $\boxed{M + 1 + n(n + p)}$ 0255
(0074) = 15 $\boxed{M + 1 + n(n + p)}$ 0205
Ces deux dernières instructions participeront au blocage, à partir de la mémoire d'adresse $\boxed{M + n(n + p)}$, des (n − k) mémoires associées: aux équations n'ayant pas encore fourni de pivot, et aux inconnues restant à éliminer, après chacune des "n" éliminations.

(0090) = 61 $\boxed{M + n - 1 + n(n + p)}$ 0100
en position facteur, nous avons: la première adresse, diminuée de l'unité, de la série de (n + p) mémoires consécutives qui contiendront, pendant chaque élimination, l'équation pivot préalablement divisée par le pivot.

(0117) = 21 $\boxed{M + n + n(n + p)}$ 0304
Instruction pour le stockage des termes de l'équation pivot, divisés par le pivot, dans une série de mémoires de travail consécutives.

(0144) = oo (n + p) oooo
(0150) = 69 (M) 0153
Instruction pour l'appel des termes des "n" équations au cours de chaque élimination.

(0162) = oo (p) oooo
(0298) = oo (n) oooo

2°) <u>à contenus variables</u>.

(0003) = valeur exacte du pivot de l'élimination en cours.
(0009) = Mot d'ORIENTATION
Son contenu sera:
44 0007 0152 pour le choix du premier pivot.
44 0007 0115 dans tous les autres cas.

(0014) = oo oooo (n + p)
au tout début des calculs.

= 21 xxxx 0304

pour le stockage des termes de l'équation pivot divisés par le pivot, dans une série de mémoires de travail consécutives à partir de l'adresse $[M + n + n(n + p)]$.

Initialement, nous avons: (0014) = (0117)

= 21 xxxx 0057

pour le stockage des résultats de chaque élimination.

Au début de chaque élimination, nous avons:

xxxx = M - 1

(0015) = oo $(k_p - 1)$ oooo

"k_p" représente l'ordre actuel de l'équation pivot de la k° élimination.

Remarque

Dans le cas du premier choix de pivot, nous avons:

(0015) = oo (n) oooo

(0021) = Mot d'ORIENTATION

Son contenu sera:

20 0096 0150 pour le choix du premier pivot.

20 0096 8001 dans tous les autres cas.

(0025) = oo $(n - k + k_i - k_p + 1)$ oooo

Cette quantité représente, au cours de la k° élimination $(1 \leqslant k \leqslant n)$, le rang qui devra être attribué à chaque terme de la k_i° colonne de la matrice inverse à l'intérieur de chaque équation, dans le but d'obtenir à la fin de tous les calculs: une matrice inverse ayant toutes ses colonnes convenablement rangées.

D'autre part:

- "k_i" représente l'ordre initial de l'équation pivot de la k° élimination.
  Nous avons: $1 \leqslant k_i \leqslant n$
- "k_p" représente l'ordre actuel de la même équation. Nous avons: $1 \leqslant k_p \leqslant n - k + 1$

(0026) = 60 AdrP 0050

pour la division de l'équation pivot par le pivot.

Nous avons en adresse facteur: "AdrP" qui représente l'adresse de la mémoire contenant le pivot.

Le contenu de la mémoire 0026 sera utilisé pour éviter la division du pivot par lui-même.

= 69 xxxx 0153

pour les calculs de la k° élimination.

Ce contenu servira à tester dans chaque équation:

- le terme de même rang que le pivot.
  Nous aurons; "xxxx" = adresse de la mémoire contenant le terme de même rang que le pivot dans l'équation en cours de traitement.
- le terme de rang $(n - k + 2)$.
  Nous aurons: "xxxx" = adresse de la mémoire contenant le terme de rang $(n - k + 2)$ dans l'équation en cours de traitement.
  Cette détection permettra de limiter le choix du pivot pour la prochaine élimination, aux termes relatifs aux inconnues restant à éliminer.
- le terme de rang $(n - k + k_i - k_p + 2)$.
  Nous aurons: "xxxx" = adresse de la mémoire contenant le terme de rang $(n - k + k_i - k_p + 2)$ dans l'équation en cours de traitement.
  Ce test a pour but de permettre la mise en place, dans chaque équation, le coefficient de la k_i° colonne de la matrice inverse, introduite par la k° élimination.
- le dernier seconds mémbre.
  Nous aurons: "xxxx" = adresse, augmentée de l'unité, de la mémoire contenant le dernier second membre de l'équation en cours de traitement.

(0031) = oo $(l_p - 1)$ oooo

Au début de la k° élimination, "l_p" représente le rang actuel de l'inconnue à éliminer.

Ensuite, "l_p" représente le rang actuel de l'inconnue à éliminer au cours de l'élimination suivante.

(0033) = Mot d'ORIENTATION

Son contenu sera:

69 0203 0106 pour le choix du premier pivot.
et successivement:
65 0072 0328 et
60 0058 0215 dans tous les autres cas.

(0034) = Mot d'ORIENTATION associé aux équations
Son contenu sera successivement au cours de chaque élimination:
oo oo98 oooo
oo oo99 oooo
oo oo9o oooo

(0038) = oo $(l_p - 1)$ oooo
"l_p" représente l'ordre actuel de l'inconnue éliminée au cours de la k° élimination.
Remarque
Dans le cas du premier choix de pivot, nous avons:
(0038) = oo (n) oooo

(0039) = Au début de chaque élimination: la valeur absolue du pivot.
Ensuite, cette mémoire est remise à zéro.
Au cours de chaque élimination, cette mémoire contient la valeur absolue du pivot pour la prochaine élimination.

(0045) = oo $(n - k + l_i - l_p + 1)$ oooo
Cette quantité réprésente, au cours de la k° élimination ($1 \leqslant k \leqslant n$), le rang qui devra être attribué à l'équation pivot, provisoirement stockée dans des mémoires de travail, dans le but d'obtenir des résultats convenablement rangés.
D'autre part:
- "l_i" représente l'ordre initial de l'inconnue éliminée au cours de la k° élimination.
  Nous avons: $1 \leqslant l_i \leqslant n$
- "l_p" représente l'ordre actuel de la même inconnue. Nous avons: $1 \leqslant l_p \leqslant n - k + 1$

(0058) = oo $(n - k + 1)$ oooo
avec $1 \leqslant k \leqslant n$

(0059) = oo $(k_p - 1)$ oooo
Au début de la k° élimination, "k_p" représente le rang actuel de l'équation pivot pour cette élimination.
Ensuite, "k_p" représente le rang actuel de l'équation pivot pour l'élimination suivante.

(0064) = 69 xxxx 8003
Cette instruction participera au blocage, à partir de la mémoire d'adresse $[M + n(n + p)]$, des $(n - k)$ mémoires associées aux équations n'ayant pas encore fourni de pivot, et aux inconnues restant à éliminer, après la k° élimination.
Ce blocage se fera en deux étapes:
- blocages des quantités en "position facteur"
  Initialement, nous aurons:
  $$xxxx = M + n(n + p) + k_p - 1$$
- blocage des quantités en "position instruction suivante".
  Initialement, nous aurons alors:
  $$xxxx = M + n(n + p) + l_p - 1$$

= 60 xxxx 0050
pour l'appel des termes de l'équation pivot afin de les diviser par le pivot.
Initialement, nous avons: "xxxx" = l'adresse de la mémoire contenant le premier terme de l'équation pivot.

= 69 xxxx 0153
pour l'appel des termes de toutes les équations au cours de chaque élimination.
Nous avons initialement: xxxx = M

(0068) = comptage des équations au cours de chaque élimination.
Ce compteur est remis à zéro au début de chaque élimination.

(0080) = Mot d'ORIENTATION
Ce mot est utilisé pendant la division de l'équation pivot par le pivot, il contient successivement:
60 0064 0269
10 8001 0091

(0084) = Mot d'ORIENTATION associé aux termes d'une même équation.
Son contenu sera successivement au cours du traitement de chaque équation:
oo oo88 oooo
oo oo89 oooo
oo oo9o oooo

(0096) = 61 xxxx 0100
Instruction d'appel des coefficients de l'équation pivot au cours de chaque élimination et pour le traitement de chaque équation. Il s'agit de l'équation pivot divisée par le pivot et stockée dans une série de mémoires de travail consécutives.
Au début du traitement de chaque équation, nous avons: (0096) = (0090)

(0103) = multiplicateur relatif à l'équation en cours de traitement.

(0115) = Mot d'ORIENTATION
Son contenu sera successivement, au cours du traitement de chaque équation:
60 8001 0268 et
60 0068 0123

(0116) = instruction de SORTIE du Sous Programme de Résolution de Systèmes Linéaires et d'Inversion de Matrice.

(0122) = oo xxxx oooo
Ce contenu servira à tester au cours de chaque élimination:
- l'équation pivot.
  Nous aurons: (0122) = (0016)
- l'équation de rang (n - k + 2).
  Il s'agit de détecter la fin du choix de pivot pour la prochaine élimination.
  Nous avons dans ce cas: (0122) = (0058)
- l'équation de rang $(n - k + l_i - l_p + 2)$.
  Il s'agit de reclasser convenablement l'équation pivot de l'élimination en cours.
  Nous aurons: (0122) = (0045)
- la fin des calculs.

Nous aurons dans ce dernier cas: (0122) = (0298)

(0250) = 21 AdrP 0057
Au début de chaque élimination, "AdrP" représente l'adresse de la mémoire contenant la valeur exacte du pivot pour l'élimination en cours.
Ensuite, "AdrP" représente l'adresse de la mémoire contenant la valeur exacte du pivot pour l'élimination suivante.

INSTRUCTIONS

Initialisation.

0000	24 0116 0322	(0116) = instruction de sortie
0322	69 8003 0234	
0234	23 0014 0173	(0014) = oo oooo (n + p)
0173	69 0058 0261	(0058) = oo xxxx oooo
0261	22 0298 0201	(0298) = oo (n) oooo
0201	24 0016 0223	(0016)
0223	24 0038 0191	(0038)
0191	24 0058 0311	(0058)
0311	35 0004 0273	
0273	22 0144 0297	(0144) = oo (n + p) oooo
0297	16 0298 0207	(0298)
0207	22 0162 0265	(0162) = oo (p) oooo
0265	65 8003 0323	(8003) = oo (M) oooo

car $n_{Max.} = 36$

```
0323    21 0068 0224        (0068) = oo oooo oooo
0224    24 0039 0292        (0039)
0292    69 0150 0257        (0150) = 69 xxxx 0153
0257    22 0150 0307        (0150) = 69 ( M ) 0153
0307    10 0014 0274        (0014) = oo oooo (n + p)
0274    19 0298 0324        (0298) = oo ( n ) oooo
                            (8003) = oo ( M ) oooo
                            (8002) = oo (n(n + p)) oooo
0324    15 8003 0284        (8002) = oo [M + n(n + p)] oooo
```

en position facteur, la pre - mière adresse de la série de "n" mémoires utilisées pour la mise en ordre des résultats.

```
0284    15 0238 0246        (0238) = 15 0001 0205
0246    20 0074 0235        (0074) = 15 [M + 1 + n(n + p)] 0205
0235    15 0288 0296        (0288) = oo oooo oo5o
0296    20 0046 0300        (0046) = 15 [M + 1 + n(n + p)] 0255
0300    15 0258 0315        (0258) = 49 9998 9900
0315    20 0044 0251        (0044) = 65 [M + n(n + p)] 0155
0251    15 0308 0166        (0308) = - oo oooo oo5o
0166    20 0022 0325        (0022) = 65 [M + n(n + p)] 0105
```

Chargement des mémoires utilisées pour la mise en ordre des résultats de chaque élimination.
Au début des calculs, la mémoire d'adresse $[M + n(n + p) + j - 1]$ est associé à la j° inconnue et à la j° équation. Elle sera chargée avec:

$$oo\ (j - 1)\ (j - 1)$$

Les quantités en "position facteur" seront utilisées pour la mise en ordre des colonnes de la matrice inverse; celles en "position instruction suivante" seront utilisées pour la mise en ordre des inconnues.

```
        0325    65 8001 0285
        0285    15 0289 8002        (0289) = - 43 9999 9896
(1)     8002    21 xxxx 0209
        0209    11 0298 0259        (0298) = oo ( n ) oooo
        0259    46 0262 0216
```

Le chargement des "n" mémoires n'est pas terminé.

```
        0262    10 8001 0240
        0240    10 0212 0126        (0212) = oo oool oool
        0126    15 0072 8002  (1)   (0072) = oo oool oooo
```

Le chargement des "n" mémoires est terminé.

```
        0216    15 0176 0290        (0176) = oo oooo oo95
        0290    20 0117 0226        (0117) = 21 [M + n + n(n + p)] 0304
```

en position facteur, la pre - mière adresse d'une série de (n + p) mémoires de travail utilisées pour le stockage de l'équation pivot au cours de chaque élimination.

```
        0226    15 0241 0301        (0241) = 39 9998 9796
        0301    20 0090 0002        (0090) = 61 [M + n - 1 + n(n + p)] 0100
        0002    15 0144 0052        (0144) = oo (n + p) oooo
        0052    22 0010 0266        (0010) = oo [M + n - 1 + (n+1)(n+p)] oooo
```

Préparation de la recherche du premier pivot.

```
        0266    69 0276 0291        (0276) = 44 0007 0152
        0291    24 0009 0312
        0312    69 0316 0326        (0316) = 20 0096 0160
        0326    24 0021 0102
        0102    69 0200 0060        (0200) = 69 0203 0106
        0060    24 0033 0254  (2)
```

Maintenant la recherche du premier pivot peut s'effectuer. Cette recherche présente un aspect particulier. EN effet, et celà dès la première élimination, nous ferons le choix du pivot pour la (k + 1)° élimination en même temps que nous ferons les calculs de la k° élimination.

Nous allons par conséquent exposer la suite du programme dans le cas général en mettant en évidence les embranchements particuliers à la recherche du premier pivot.

k° ELIMINATION

Au début de cette élimination, nous avons "n" équations formées chacune de (n + p) termes, qui se répartissent de la façon suivante:

- (n - k + 1) termes, qui forment le premier membre.
- (k - 1 + p) termes, qui constituent les seconds membres. Parmi ceux-ci, nous avons:
  - les (k - 1) premiers termes, qui appartiennent aux colonnes de la matrice inverse de la matrice du système. Ces colonnes ont été introduites une à une, lors des (k - 1) éliminations précédentes.
  - les "p" derniers termes, qui sont issus des "p" seconds membres "donnés".

Les termes d'une même équation sont enregistrés dans des mémoires consécutives. Les premiers termes des "n" équations sont, et resteront toujours, dans les mémoires d'adresses: M , M+(n+p) , M+2(n+p) , ... , M+(n-1)(n+p).

Le pivot, qui a été choisi en même temps que s'effectuaient les calculs de l'élimination précédente (mis à part le cas du premier pivot), est caractérisé par:

- sa valeur absolue: contenu de la mémoire 0039.
- son adresse: en position facteur du contenu de la mémoire d'adresse 0250.
- " k_p ": le numéro d'ordre de l'équation dans laquelle il a été trouvé.

  (0059) = oo (k_p - 1) oooo
- " l_p ": le numéro de l'inconnue dont il est le coefficient.

  (0031) = oo (l_p - 1) oooo

Notons que nous aurons toujours k_p et l_p inférieurs ou égaux à (n - k + 1).

La k° élimination, qui porte sur les (n + p) termes de chacune des "n" équations, va donc permettre:

1°) L'annulation du terme de rang l_p dans chacune des (n - 1) équations qui forment, avec l'équation pivot (la k_p° équation), le système linéaire d'ordre "n".
Si l'on appelle " l_i ": l'ordre initial des termes de rang l_p , nous dirons que la k° élimination a éliminé la l_i° inconnue.

2°) L'introduction d'une nouvelle colonne de la matrice inverse du système, qui n'avait pas été affectée par les (k - 1) éliminations précédentes.
Si l'on appelle " k_i ": l'ordre initial de la k_p° équation, la colonne de la matrice inverse introduite sera la k_i°. Elle est issue de la colonne de même rang de la "matrice unité". Par conséquent, tous les coefficients de cette colonne de la matrice unité seront nuls, sauf celui situé sur la k_i° ligne, qui sera égal à l'unité.

Les résultats de la k° élimination devront être rangés de façon à pouvoir donner toute leur signification aux résultats définitifs. Celà sera, si nous reclassons convenablement l'équation pivot à la suite des (n - k) équations n'ayant pas encore fourni de pivot; et si nous plaçons correctement la k_i° colonne de la matrice inverse parmi les (k - 1) colonnes déjà conservées en mémoires.

Au début de la k° élimination, nous avons (n - k + 1) mémoires consécutives à partir de l'adresse $[M + n(n + p)]$, qui sont associées aux équations n'ayant pas encore fourni de pivot et aux inconnues restant à éliminer.
Par exemple: la quantité en position facteur de la j° mémoire (adresse: $[M+n(n+p)+j-1]$) indique l'ordre initial de la j° équation actuelle; et la quantité en position instruction suivante indique l'ordre initial de la j° inconnue actuelle.

Les contenus de ces (n - k + 1) mémoires vont nous permettre un rangement correct des résultats de la k° élimination.

En effet, la quantité en position "facteur" du contenu de la mémoire d'adresse $[M+n(n+p)+k_p-1]$ nous donne l'ordre initial " k_i " de la k_p° équation.

$$(M+n(n+p)+k_p-1) = oo\ (k_i - 1)\ xxxx$$

La quantité $(k_i - k_p)$, toujours positive, représente le nombre d'équations d'ordres initiaux inférieurs à k_i , qui ont déjà fourni un pivot. Par suite, la quantité $(k_i - k_p)$ représente aussi le nombre de colonnes de la matrice inverse du système, d'ordres inférieurs à k_i , qui ont déjà été introduites. En conséquence, dans chaque équation, le coefficient de la k_i° colonne de la matrice inverse sera placé:

- après les (n - k) termes restant au premier membre,
- et après les (k_i-k_p) coefficients des colonnes d'ordres inférieurs à k_i .

C'est à dire que le coefficient de la k_i° colonne de la matrice inverse occupera le rang:

$$n - k + k_i - k_p + 1$$

De la même façon, la quantité en position "instruction suivante" du contenu de la mémoire d'adresse $[M+n(n+p)+l_p-1]$ nous donne l'ordre initial " l_i " de la l_p° inconnue.

$$(M+n(n+p)+l_p-1) = oo\ xxxx\ (l_i - 1)$$

La quantité $(l_i - l_p)$, toujours positive, représente le nombre d'inconnues précédant la l_i°, qui ont déjà été éliminées. Les équations pivots correspondantes devront précéder l'équation pivot à classer. Par conséquent, cette équation sera précédée au total de:

$$(n-k+l_i-l_p)\ \text{équations.}$$

Son premier terme sera à enregistrer dans la mémoire d'adresse:

$$M + (n - k + l_i - l_p)(n + p)$$

Finalement, avant de commencer les calculs de la k° élimination, nous bloquerons dans (n - k) mémoires consécutives à partir de l'adresse M+n(n+p) , les renseignements relatifs:aux équations n'ayant pas encore fourni de pivot et aux inconnues restant à éliminer, après la k° élimination. A ce moment, la situation sera absolument identique à ce qu'elle était au début de la k° élimination.

En procédant de cette manière, à la fin des "n" éliminations, nous aurons "n" équations composées chacune de (n + p) termes qui se répartiront de la façon suivante:

- aucun terme pour le premier membre,
- (n + p) termes qui constituent les seconds membres. Parmi ceux-ci, nous aurons:
  - les "n" premiers termes qui seront les coefficients des "n" colonnes, convenablement rangées, de la matrice inverse du système.
  - les "p" derniers termes qui seront les racines de même rang, correspondant à chacun des "p" seconds membres "donnés".

Par conséquent, nous trouverons les "n" coefficients de la j° colonne de la matrice inverse, dans les mémoires d'adresses:

M + j - 1
M + (n+p) + j - 1
.....
M + (n-1)(n+p) + j - 1

et de la même façon, nous trouverons les "n" racines, convenablement rangées, relatives à la i° série de seconds membres dans les mémoires d'adresses:

M + n + i - 1
M + n + (n+p) + i - 1
.....
M + n + (n-1)(n+p) + i - 1

Restauration des aiguillages.

```
(13)  0200   69 0203 0106        (0203) = 44 0007 0115
      0106   24 0009 0012
      0012   69 0015 0018        (0015) = 20 0096 8001
      0018   24 0021 0024
      0024   69 0027 0030        (0027) = 65 0072 0328
      0030   24 0033 0036
      0036   60 0039 0043        (0039) = valeur absolue du
                                          pivot.
      0043   44 0047 0048
```

Le pivot est nul.

```
      0048   01 0000 9999   La matrice du système est singulière.
                            On ne peut pas poursuivre les calculs.
```

Le pivot n'est pas nul.

```
      0047   65 0250 0055   (0250) = 21 AdrP 0057
                            AdrP = Adresse du Pivot
      0055   15 0008 8002   (0008) = 48 0000 0196
      8002   69 AdrP 0253
      0253   24 0003 0156   (0003) = valeur exacte du
                                    pivot.
      0156   65 0059 0013   (0059) = oo (k_p - 1) oooo
      0013   24 0016 0019   (0016)
      0019   15 0022 8002   (0022) = 65 [M + n(n + p)] 0105
      8002   65 yyyy 0105   yyyy = M + n(n + p) + k_p - 1
                            (8002) = oo (k_i - 1) (l'_i - 1)
      0105   15 0058 0063   (0058) = oo (n - k + 1) oooo
      0063   16 0059 0113   (0059)
      0113   21 0068 0071   (0068) = oo oooo oooo
                            Remise à zéro du compteur des
                            équations.
      0071   22 0025 0028   (0025) = oo (n-k+k_i-k_p+1) oooo
      0028   65 0031 0035   (0031) = oo (l_p - 1) oooo
```

0035	24 0038 0041	(0038) = oo $(l_p - 1)$ oooo
0041	15 0044 8002	(0044) = 65 $\left[M + n(n + p)\right]$ 0155
8002	65 zzzz 0155	zzzz = $M + n(n + p) + l_p - 1$
		(8002) = oo $(k'_i - 1)$ $(l_i - 1)$
0155	35 0004 0065	
0065	15 0058 0163	(0058) = oo $(n - k + 1)$ oooo
0163	16 0031 0085	(0031) = oo $(l_p - 1)$ oooo
0085	21 0039 0042	(0039) = oo oooo oooo
		Remise à zéro de la mémoire contenant la valeur maximum du pivot, en vue du prochain choix de pivot.
0042	22 0045 0098	(0045) = oo $(n-k+l_i-l_p+1)$ oooo

Blocage, à partir de la mémoire d'adresse $\left[M + n(n + p)\right]$, des $(n - k)$ mémoires associées: aux équations n'ayant pas encore fourni de pivot, et aux inconnues restant à éliminer, après la k° élimination.

Au cours de cette élimination, l'équation d'ordre k_p est "équation pivot". Elle permet l'élimination de l'inconnue d'ordre l_p (ordre initial: l_i) et l'introduction de la "k_i° " colonne de la matrice inverse. Cette "k_p° " équation sera reclassée à la suite des $(n - k)$ premières équations en fonction de l'inconnue qu'elle aura servie à éliminer. Par conséquent, toutes les équations d'ordre supérieur à "k_p", qui n'ont pas encore fourni de pivot, vont gagner une position. Il faut donc que les quantités,en position facteur des mémoires associées à ces équations, subissent un déplacement analogue pour qu'elles conservent toute leur signification.
En conséquence, nous commencerons à travailler sur la position facteur du contenu de la mémoire associée à la $(k_p + 1)$° équation. Son adresse est égale à $\left[M + n(n + p) + k_p\right]$.

0098	60 0016 0121	(0016) = oo $(k_p - 1)$ oooo
0121	10 0074 8003	(0074) = 15 $\left[\begin{matrix}M + 1 \\ + n(n + p)\end{matrix}\right]$ 0205
		(8003) = 15 $\left[\begin{matrix}M + k_p \\ + n(n^p + p)\end{matrix}\right]$ 0205
(14) 8003	15 xxxx 0205	

0205	10 0108 0213	(0108) = 53 9999 7798
0213	21 0064 0017	(8003) = 69 wwww 8003
		Dans le cas présent, nous vérifierons: wwww = xxxx - 1.
0017	11 0020 0064	(0020) = 47 0000 7700
		(8003) = 22 wwww 0303
0064	69 wwww 8003	
8003	22 wwww 0303	
0303	11 0058 0263	(0058) = oo $(n - k + 1)$ oooo
0263	11 0074 0029	(0074) : voir page précédente.
0029	10 0032 0037	(0032) = 02 0002 9902
		(8003) = 09 dddd 0000
		avec:
		dddd = wwww - $\left[M+n(n+p)+n-k-1\right]$
		suivant le signe de "dddd", nous aurons soit un "8", soit un "9" en position 9 du contenu de 8003.
0037	69 8003 0094	
0094	99 0148 0099	

Le blocage des quantités en position facteur, n'est pas terminé.

0148	10 0058 0313	(0058)
0313	10 0074 0079	(0074)
0079	11 0082 0087	(0082) = 09 0001 0000
		(8003) = 15 (xxxx + 1) 0205
0087	16 8002 8003 (14)	

Le blocage des quantités en position facteur,est terminé.
Nous allons maintenant effectuer le blocage des quantités en position instruction suivante.

Au cours de la k° élimination, l'inconnue d'ordre l_p disparait. Par conséquent, tous les coefficients des inconnues d'ordres supérieurs à l_p, qui restent à éliminer, vont gagner une position. Il faut donc que les quantités en position instruction suivante, associées à ces inconnues, subissent un déplacement analogue

pour qu'elles conservent toute leur signification.
En conséquence, nous commencerons à travailler sur la position instruction suivante du contenu de la mémoire associée à la $(l_p + 1)^e$ inconnue. Son adresse est égale à $\left[M + n(n + p) + l_p\right]$.

	0099	60 0038 0093	(0038) = oo $(l_p - 1)$ oooo
	0093	10 0046 8003	(0046) = 15 $\left[M + 1 + n(n + p)\right]$ 0255
			(8003) = 15 $\left[M + 1 + n(n^p + p)\right]$ 0255
(3)	8003	15 xxxx 0255	
	0255	10 0158 0114	(0158) = 53 9999 7748
	0114	21 0064 0067	(8003) = 69 wwww 8003
			Dans le cas présent, nous vérifierons encore:
			wwww = xxxx - 1
	0067	11 0070 0064	(0070) = 46 0000 7849
			(8003) = 23 wwww 0154
	0064	69 wwww 8003	
	8003	23 wwww 0154	
	0154	11 0058 0164	(0058) = oo (n - k + 1) oooo
	0164	11 0046 0001	(0046)
	0001	10 0204 0109	(0204) = 01 0003 0101
			(8003) = 09 dddd 0000
			avec:
			dddd = wwww - $\left[M+n(n+p)+n-k-1\right]$
			suivant le signe de "dddd", nous aurons: soit un "8", soit un "9" en position 9 du contenu de 8003.
	0109	69 8003 0066	
	0066	99 0120 0171	

Le blocage des quantités en position instruction suivante, n'est pas terminé.

0120	10 0023 0148	(0023) = 00 0000 0050
0148	10 0058 0313	(0058)
0313	10 0074 0079	(0074) = voir page précédente.
0079	11 0082 0087	(0082) = 09 0001 0000
		(8003) = 15 (xxxx + 1) 0255
0087	16 8002 8003 (3)	

Le blocage des quantités en position instruction suivante, est terminé.
Transfert de l'équation pivot dans des mémoires de travail.

L'équation pivot est formée de (n + p) termes situés dans des mémoires consécutives à partir de l'adresse $\left[M + (k_p - 1)(n + p)\right]$. Nous trouvons dans l'ordre:

- les (n - k + 1) coefficients des inconnues restant à éliminer; parmi ceux-ci: le k_p^e est le pivot.
- les (k - 1) coefficients, convenablement classés, des colonnes de la matrice inverse introduites lors des éliminations précédentes.
- enfin, les "p" seconds membres.

Nous allons diviser ces (n + p) termes par le pivot, situé dans la mémoire d'adresse:

$$AdrP = M + (k_p - 1)(n + p) + l_p - 1$$

La division du pivot par lui-même donne l'unité. Nous n'effectuerons pas ce calcul, et nous ne mettrons pas le résultat correspondant en mémoire.

Aux (n + p - 1) autres résultats, nous ajouterons le coefficient de la k_i^e colonne de la matrice inverse introduite par la k^o élimination. Ce coefficient, égal à l'inverse du pivot, sera placé en $(n - k + k_i - k_p + 1)^e$ position.

A la fin des calculs, du fait de la disparition du pivot et de l'introduction du coefficient de la k_i^e colonne de la matrice inverse, nous aurons encore (n + p) termes qui seront mis dans des mémoires consécutives de travail à partir de l'adresse $\left[M + n(n + p) + n\right]$.

0171	69 0124 0077	(0124) = 60 0064 0269
0077	24 0080 0083	(0080)

```
0083    60 0250 0305        (0250) = 21 AdrP 0057
0208    10 0208 0214        (0208) = 38 9999 9993
0214    15 0117 0221        (0117) = 21 [M + n
                                        + n(n + p)] 0304
0221    21 0026 0129        (0026) = 60 AdrP 0050
0129    11 0038 0143        (0038) = oo (l_p - 1) oooo
                            (8003) = 60 (AdrP-l_p+1) 0050
```

en position facteur, l'adresse du premier coefficient de l'équation pivot.

(8002) = 21 $\begin{bmatrix} M + n \\ + n(n + p) \end{bmatrix}$ 0304

en position facteur, la pre - mière adresse d'une série de (n + p) mémoires de travail.

```
(4)  0143    21 0064 0167        (0064)
     0167    20 0014 0217        (0014)
     0217    11 0026 0081        (0026)
     0081    44 0135 0086
```

Le prochain terme à diviser est le pivot. Nous passons à la division suivante sans effectuer d'opération.

```
0086    60 0064 0169        (0064)
0169    15 0014 0219        (0014)
0219    10 0072 0143   (4)   (0072) = oo oool oooo
```

Le prochain terme à diviser n'est pas le pivot.
Avant d'effectuer la division, il faut s'assurer que le terme à diviser n'est pas de rang égal à $(n - k + k_i - k_p + 2)$

```
0135    10 0038 0193        (0038)
0193    11 0025 0179        (0025) = oo (n-k+k_i-k_p+1) oooo
```

A ce moment, nous avons en position facteur du contenu de 8003 : l'adresse du prochain terme à diviser, moins celle du premier terme de l'équation pivot, moins la quantité $(n - k + k_i - k_p + 1)$.

```
0179    44 0133 0080
```

Le prochain terme à diviser n'est pas de rang $(n-k+k_i-k_p+2)$.
Avant d'effectuer la division, vérifions que le rang du terme à diviser est inférieur à (n + p + 1).

```
        0133    10 8001 0091
(6)     0091    11 0144 0149        (0144) = oo (n + p) oooo
```

A ce moment, nous avons en position facteur du contenu de 8003 : l'adresse du prochain terme à diviser, moins celle du premier terme de l'équation pivot, moins (n + p).

```
        0149    44 0064 0254        Test de fin des divisions.
```

Le rang du terme à diviser est inférieur à (n + p + 1).
Nous allons effectuer la division.

```
0064    60 xxxx 0050
```

Séquence d'ENTREE dans l'A.V.F. pour effectuer une division.

```
(5)  0050    69 0003 0006        (0003) = diviseur.
     0006    24 1801 0004
     0004    69 1900 0053        VDV
     0053    24 1808 0011
     0011    69 0014 1800        (0014) = instruction de sortie.
```

```
1800
   Sous-Programme
     A. V. F.
              1803
```

(1803) = (0014)
= 21 xxxx 0304

Mise en mémoire du résultat.

```
1803    21 xxxx 0304
```

Préparation de la division suivante.

```
0304    60 0064 0069        (0064)
0069    15 0014 0119        (0014)
```

```
0119    15 0072 0127          (0072) = oo oool oooo
0127    10 8001 0143    (4)
```

Le prochain terme à diviser est de rang $(n-k+k_i-k_p+2)$.
Il faut distinguer deux cas.

1°) C'est la première fois que la réponse à ce test est négative.
Dans ce cas, nous ajournons la division du terme de rang $(n-k+k_i-k_p+2)$ par le pivot, pour mettre en place le coefficient de la k_i° colonne de la matrice inverse.
Nous obtiendrons ce coefficient en calculant l'inverse du pivot.

```
0080    50 0064 0269          (0064)
0269    11 0072 0177          (0072)
0177    21 0064 0267          (0064)
0267    69 0133 0136          (0133) = 10 8001 0091
0136    24 0080 0183          (0080)
0183    60 0186 0050    (5)   (0186) = 51 1000 0000
```

2°) C'est la seconde fois que la réponse à ce test est négative.
Dans ce cas, nous venons juste de calculer et de mettre en place le coefficient de la k_i° colonne de la matrice inverse. Maintenant, nous allons effectuer la division du terme de rang $(n-k+k_i-k_p+2)$ par le pivot. Mais avant d'effectuer l'opération, nous allons vérifier que nous avons:

$$(n-k+k_i-k_p+2) < (n+p+1)$$

```
0080    10 8001 0091    (6)
```

Toutes les divisions sont terminées. L'équation pivot modifiée a été entièrement tranférée dans la série de "n" mémoires consécutives de travail débutant à l'adresse M+n(n+p)+n .
Nous sommes prêts à commencer les calculs d'élimination.
Pour ces calculs, nous devrons répondre aux impératifs suivants:

1°) ne faire aucun calcul avec l'équation pivot.
Il s'agit de la k_p° équation.

2°) limiter le choix de pivot pour l'élimination suivante aux seules équations n'en ayant pas encore fourni.
Ceci s'adresse aux (n-k+1) premières équations sauf la k_p°, qui est l'équation pivot de l'élimination en cours.

3°) reclasser l'équation pivot de l'élimination en cours.
Elle sera à reclasser derrière les (n - k) équations précédentes, en position de $(n-k+l_i-l_p+1)$° équation.

4°) détecter la fin de l'élimination.

Ces quatre impératifs feront l'objet d'un test situé dans la mémoire 0227.

(0227) = 44 0181 0033

Les trois premiers dépendront, après le test, d'un aiguillage du type 88, 89, 90 (contenus successifs de la mémoire 0034)
Le quatrième sera exécuté par la modification du contenu de la mémoire 0033.

5°) à l'intérieur de chaque équation, repérer le terme de même rang que le pivot.
Il s'agit du l_p° terme.

6°) à l'intérieur de chaque équation, tester la fin du premier membre ((n-k+1) termes).

7°) à l'intérieur de chaque équation, tester le terme de rang $(n-k+k_i-k_p+2)$, afin de calculer et de mettre en place le coefficient de la k_i° colonne de la matrice inverse introduite par l'élimination en cours.
Nous verrons en temps utile, comment effectuer le calcul de coefficient qui occupera le rang $(n-k+k_i-k_p+1)$.

8°) détecter la fin de chacune des équations.

Ces quatre derniers impératifs feront l'objet d'un test situé dans la mémoire de travail 0009.

(0009) = 44 0007 0115

Les 5°, 6°, 7° impératifs dépendront, après le test, d'un aiguillage du type 88, 89, 90 (contenus successifs de la mémoire 0084).

Le 8° impératif sera éxécuté par la modification du contenu de la mémoire 0115.

Signalons aussi les contenus de diverses mémoires de travail, au début de ces calculs.

1°) Dans le cas général

(0009) = 44 0007 0115
(0021) = 20 0096 8001
(0033) = 65 0072 0328
(0039) = 00 0000 0000
(0068) = 00 0000 0000
(0115) = 60 8001 0268

2°) Dans le cas du choix du premier pivot.

(0009) = 44 0007 0152
(0021) = 20 0096 0160
(0033) = 69 0203 0106
(0039) = 00 0000 0000
(0068) = 00 0000 0000
(0016) = 00 (n) 0000
(0038) = 00 (n) 0000

Enfin, sous forme de "REMARQUES" nous indiquerons les embranchements particuliers à la recherche du premier pivot.

(2)	0254	65 0150 0206	(0150) = 69 (M) 0153

REMARQUE C'est par cette instruction que commence le choix du premier pivot.

0206	24 0064 0317	(0064)
0317	69 0016 0319	(0016) = oo $(k_p - 1)$ oooo dans le cas général. oo (n) oooo dans le cas du choix du premier pivot.
0319	24 0122 0075	(0122)
0075	69 0078 0131	(0078) = oo oo88 oooo
0131	24 0034 0137	(0034)
0137	15 0040 0095	(0040) = - 48 0001 0096
		(8002) = 21 (M) 0057
(7) 0095	20 0014 0118	(0014)
(8) 0118	11 0122 0227	
0227	44 0181 0033	Test relatif aux équations.

L'équation que nous devons traiter, n'est:
- ni l'équation pivot
- ni la $(n-k+1)^e$ équation.
- ni la $(n-k+l_i-l_p+1)^e$ équation.

Les "n" équations du système n'ont pas été toutes traitées.

0181	65 0064 0170	(0064)
0170	15 0038 0243	(0038) = oo $(l_p - 1)$ oooo dans le cas général. oo (n) oooo dans le cas du choix du premier pivot.
0243	20 0026 0229	(0026) = 69 xxxx 0153
0229	15 0132 8002	(0132) = 00 0000 0103
8002	69 yyyy 0256	yyyy = adresse du terme de même rang que le pivot dans l'équation traitée.

	0256	24 0103 0306	(0103) = multiplicateur pour les calculs d'élimination relatifs à une équation.
	0306	69 0159 0062	(0159) = 60 8001 0268
	0062	24 0115 0168	(0115)
	0168	69 0078 0231	(0078) = oo oo88 oooo
	0231	24 0084 0187	(0084)
	0187	69 0090 0293	(0090) = 61 $\begin{bmatrix} M + n - 1 \\ + n(n + p) \end{bmatrix}$ 0100
	0293	24 0096 0199	(0096) en position facteur, nous avons l'adresse, diminuée de l'unité, du premier terme de l'équation pivot: divisée par le pivot et transférée dans des mémoires de travail.
(9)	0199	60 0064 0220	(0064)
	0220	11 0026 0009	0009 Mémoire de travail utilisée pour l'orientation du programme. Pour l'étude du choix du premier pivot, se reporter à la REMARQUE N°1. Nous poursuivons l'étude du programme dans le cas général.
	0009	44 0007 0115	Test relatif aux termes d'une même équation.

Le terme sur lequel nous devons opérer n'est pas:

- de même rang que le pivot
- le terme de rang (n-k+2). Fin du traitement du premier membre.
- le terme de rang ($n-k+k_i-k_p+2$).
- le dernier terme de l'équation complète.

(11)	0007	60 0014 0270	(0014)
	0270	15 0096 0051	(0096)
	0051	10 0072 0277	(0072) = oo oool oooo
	0277	15 8001 0185	(8001) = oo oool oooo
	0185	21 0014 0021	0021 Mémoire de travail utilisée pour l'orientation du programme. Pour l'étude du choix du premier pivot, se reporter à la REMARQUE N°2. Nous poursuivons l'étude du programme dans le cas général.
	0021	20 0096 8001	
	8001	61 xxxx 0100	

Séquence d'ENTREE dans l'A.V.F. pour effectuer une multiplication.

0100	69 0103 0056	(0103) = multiplicateur
0056	24 1801 0054	
0054	69 1996 0049	VMU
0049	24 1808 0061	
0061	69 0064 1800	

1800	
Sous-Programme A. V. F.	
	1803

(1803) = (0064)
= 69 xxxx 0153

Séquence d'ENTREE dans l'A.V.F. pour effectuer une addition.

1803	69 xxxx 0153	
0153	24 1801 0104	
0104	69 1902 0005	VAA
0005	24 1808 0111	
0111	69 0014 1800	

1800	
Sous-Programme A. V. F.	
	1803

(1803) = (0014)
= 21 xxxx 0057

Mise en mémoire du résultat.

1803	21 xxxx 0057	
0057	69 0034 0237	(0034) = oo oo8/9x oooo
0237	96 0140 0092	

L'équation en cours de traitement fait partie des (n-k+1) premières équations qui doivent participer au choix de pivot pour la prochaine élimination.

```
0140    69 0084 0287        (0084) = oo oo8/9x oooo
0287    96 0190 0092
```

Nous travaillons sur le premier membre d'une équation ((n-k+1) termes), le choix de pivot doit avoir lieu.

```
0190    67 8003 0097
0097    16 0039 0194        (0039) = valeur absolue du
                            pivot actuellement retenu.
0194    46 0092 0198
```

Le dernier résultat obtenu est le maximum en valeur absolue et doit être retenu comme pivot.

```
0198    15 8001 0107
0107    20 0039 0142        (0039) = nouvelle valeur
                            absolue du pivot actuellement
                            retenu.
0142    65 0068 0073        (0068) = compteur d'équations.
0073    69 0034 0088        (0034) = oo oox8/9 oooo
0088    95 0141 0244
0244    16 0072 0141        (0072) = oo oool oooo
0141    20 0059 0112        (0059) = en position facteur;
                            le numéro d'ordre, diminué de
                            l'unité, de l'équation ayant
                            fourni le pivot.
0112    65 0096 0101        (0096) = 61 xxxx 0100
0101    16 0090 0145        (0090) = 61 [M + n - 1 + n(n + p)] 0100
0145    16 0072 0327        (0072)
0327    20 0031 0134        (0031) = en position facteur,
                            le rang, diminué de l'unité,
                            du pivot retenu à l'intérieur
                            de son équation.
0134    69 0014 0218        (0014)
0218    24 0250 0092        (0250) = en position facteur,
                            l'adresse de la mémoire qui
                            contient la valeur exacte du
                            terme retenu comme pivot.
```

Le choix du pivot est terminé.
Il n'y a pas eu de choix de pivot.
Dans les deux cas: préparation du calcul suivant.

```
(10)  0092    60 0064 0320        (0064)
      0320    10 0072 0128        (0072) = oo oool oooo
      0128    21 0064 0199  (9)
```

Réponse négative au test: 44 0007 0115
La série de calculs en cours, sur une partie d'équation, est terminée.

Nous avons deux possibilités suivant le contenu de la mémoire 0115. Nous allons les étudier en considérant l'enchaînement normal des calculs.

— 1°) Nous sommes prêts à travailler sur le terme de l'équation:
- de même rang que le pivot.
- de rang (n-k+2). Fin du traitement du premier membre.
- de rang $(n-k+k_i-k_p+2)$

```
0115    60 8001 0268        (8001) = 69 xxxx 0153
0268    15 0072 0178        (0072)
0178    15 0084 0089        (0084) = oo oo8/98/9 oooo
0089    96 0192 0294
0192    95 0195 0147
```

Tous les termes de l'équation, de rang inférieur à celui du pivot, ont subis les calculs d'élimination.
Préparation d'une nouvelle série de calculs, qui portera sur tous les autres termes du premier membre.

```
0195    11 0038 0245        (0038) = oo (1_p - 1) oooo
```

0245	10 0058 0264		(0058) = oo (n - k + 1) oooo
0264	21 0026 0279		(0026) = 69 xxxx 0153 en position facteur, l'adresse, augmentée de l'unité, du dernier terme du premier membre de l'équation.
0279	20 0084 0092	(10)	(0084) = oo oo89 oooo

REMARQUE En 0092, nous modifions l'instruction d'appel des coefficients de l'équation, de façon à ne faire aucun calcul avec le terme de même rang que le pivot.

Par contre, nous ne modifions pas l'instruction de mise en mémoire des résultats.

Tout le premier membre de l'équation a subi les calculs d'élimi - nation.
Préparation , pour cette équation, d'une nouvelle série de calculs qui s'arrêtera juste avant le traitement du terme de rang $(n-k+k_i-k_p+2)$.
Nous aurons un 9 en position 6 du contenu de 0084, d'où la suppression du choix de pivot.

0147	11 0058 0314		(0058)
0314	10 0025 0329		(0025) = oo $(n-k+k_i-k_p+1)$ oooo
0329	21 0026 0130		(0026) = 69 xxxx 0153 en position facteur, l'adresse du terme de rang $(n-k+k_i-k_p+2)$ dans l'équation en cours de traitement.
0130	20 0084 0199	(9)	(0084) = oo oo9o oooo

Nous sommes prêts à travailler avec le terme de rang $(n-k+k_i-k_p+2)$ dans l'équation en cours de traitement.
Il faut maintenant calculer et mettre en place le coefficient de la $k_i°$ colonne de la matrice inverse introduite par l'élimination en cours; et préparer le test de fin des calculs pour l'équation en cours de traitement.

Du fait de l'élimination de l'inconnue de rang l_p (rang initial: l_i) dans l'équation en cours de traitement, le résultat obtenu avec le terme de rang $(n-k+k_i-k_p+1)$ occupe maintenant le rang $(n-k+k_i-k_p)$. Ceci est, et sera, d'ailleurs vrai pour les résultats obtenus à partir de tous les termes de rang supérieur à "l_p" et inférieur à $(n-k+k_i-k_p+2)$.
Par suite nous disposons de la mémoire ayant contenu le terme de rang $(n-k+k_i-k_p+1)$, pour calculer et stocker le coefficient de la $k_i°$ colonne de la matrice inverse introduite par l'élimination en cours.
Le calcul de ce coefficient entre dans le cadre général des calculs d'élimination; il suffit de travailler sur la valeur initiale de ce coefficient, que nous savons être nulle.

0294	11 0025 0180	(0025) = oo $(n-k+k_i-k_p+1)$ oooo
0180	10 0144 0249	(0144) = oo (n + p) oooo
0249	21 0026 0230	(0026) = 69 xxxx 0153 en position facteur, l'adresse, augmentée de l'unité, du dernier terme de l'équation en cours de traitement; ou celle du premier terme de l'équation suivante.
0230	69 0233 0236	(0233) = 60 0068 0123
0236	24 0115 0318	(0115) : Modification de la réponse négative au "Test relatif aux termes d'une même équation".
0318	65 0064 0271	(0064) = 69 xxxx 0153 en position facteur, l'adresse du terme de rang $(n-k+k_i-k_p+2)$
0271	16 0072 0228	(0072) = oo oool oooo
0228	20 0064 0321	(0064)
0321	15 0174 8002	(0174) = - 48 0000 0146
8002	21 xxxx 0007	(8003) = oo oooo oooo Mise en place de la valeur initiale du coefficient de la $k_i°$ colonne de la matrice inverse.

```
0007                              (11)  Retour au programme général
                                        pour le calcul du coefficient
                                        de la k_i° colonne de la matrice
                                        inverse, après la k° élimination
```

-- 2°) L'équation entière a subi les calculs de la k° élimination. Nous allons préparer le passage à l'équation suivante.

```
0115    60 0068 0123          (0068) = compteur des
                                       équations.
0123    10 0072 0278          (0072) = oo oool oooo
0278    21 0068 0118    (8)   (0068)
```

REMARQUE Etant donné que pour chaque équation, nous avons: disparition du terme de même rang que le pivot, et introduction du coefficient de la k_i° colonne de la matrice inverse; le nombre de termes de chaque équation est finalement constant et égal à (n + p).

Réponses négatives au test: 44 0181 0033
L'équation rencontrée présente une particularité. Le programme doit être ajusté ou modifié.

Nous avons deux possibilités suivant le contenu de la mémoire 0033. Nous allons les étudier en considérant l'enchaînement normal des calculs.

-- 1°) Nous sommes prêts à aborder le traitement de:
- l'équation pivot (de rang k_p)
- l'équation de rang (n-k+2).
- l'équation de rang ($n-k+l_i-l_p+2$).

```
0033    65 0072 0328          (0072) = oo oool oooo
0328    15 0034 0139          (0034) = oo oo8/98/9 oooo
0139    96 0242 0295
0242    95 0146 0197
```

L'équation que nous sommes prêts à traiter est la k_p° équation. Elle est "équation pivot" pour la k° élimination et ne doit pas subir de calcul. Nous passons à l'équation suivante après préparation du test sur la (n-k+2)° équation.

```
0146    20 0034 0138          (0034) = oo oo89 oooo
0138    69 0058 0161          (0058) = oo (n-k+1) oooo
0161    24 0122 0125          (0122)
0125    65 0064 0172          (0064) = 69 xxxx 0153
                              en position facteur, l'adresse
                              du premier terme de la k_p°
                              équation.
0172    15 0144 0299          (0144) = oo (n + p) oooo
0299    20 0064 0233          (0064)
0233    60 0068 0123          (0068) = compteur des
                                       équations.
0123    10 0072 0278          (0072) = oo oool oooo
0278    21 0068 0118    (8)   (0068)
```

L'équation que nous sommes prêts à traiter est la (n-k+2)° équation. Le choix du pivot pour l'élimination suivante doit être supprimé pour les autres équations (9 en position 6 du contenu de 0034).
Préparation du test sur la ($n-k+l_i-l_p+2$)° équation.

```
0197    20 0034 0188          (0034) = oo oo90 oooo
0188    69 0045 0248          (0045) = oo (n-k+l_i-l_p+1) oooo
0248    24 0122 0175          (0122)
0175    60 0068 0118    (8)   (0068) = compteur des
                                       équations.
```

L'équation que nous sommes prêts à traiter est la ($n-k+l_i-l_p+2$)° équation. Avant d'effectuer ce travail, il faut ranger l'équation pivot en position de ($n-k+l_i-l_p+1$)° équation, de façon à obtenir des résultats parfaitement rangés après les "n" éliminations.

Préparation du test sur la n° équation et modification du contenu de la mémoire de travail 0033.

```
0295    69 0298 0151          (0298) = oo ( n ) oooo
0151    24 0122 0225          (0122)
```

0225 69 0280 0283 (0280) = 60 0058 0215
0283 24 0033 0286 (0033)

Nous allons translater les contenus des (n + p) mémoires de travail consécutives à partir de l'adresse $[M + n(n + p) + n]$ dans des mémoires consécutives à partir de l'adresse:

$$M + (n-k+l_i-l_p)(n + p)$$

0286 60 0090 0196 (0090) = 61 $\begin{bmatrix} M + n - 1 \\ + n(n + p) \end{bmatrix}$ 0100
0196 15 0014 0222 (0014) = 21 zzzz 0057
zzzz = $M+(n-k+l_i-l_p)(n+p)-1$
0222 10 0275 0281 (0275) = 08 0000 7902
0281 15 0184 0189 (0184) = 03 0000 0100
(8002) = 24 zzzz 0157
(8003) = 69 $\begin{bmatrix} M +n - 1 \\ + n(n + p) \end{bmatrix}$ 8002
(12) 0189 10 0072 0182 (0072) = oo oool oooo
0182 15 8001 8003
8003 69 xxxx 8002
8002 24 xxxx 0157
0157 11 0010 0165 (0010) = oo $\begin{bmatrix} M + n - 1 \\ +(n+1)(n+p) \end{bmatrix}$ oooo
il s'agit de l'adresse du dernier terme à transférer.
0165 69 8003 0272
0272 99 0076 0232

La translation n'est pas terminée.

0076 10 0010 0189 (12) (0010)

La translation est terminée.
Préparation des calculs pour l'équation suivante, c'est à dire la $(n-k+l_i-l_p+2)°$ équation.

0232 16 0184 0239 (0184)
0239 11 8003 0247
0247 10 0068 0095 (7) (0068) = compteur des équations.

(8003) = (0068)
(8002) = 21 xxxx 0057
en position facteur, l'adresse, diminuée de l'unité, du premier terme de la $(n-k+l_i-l_p+2)°$ équation.

— 2°) Les "n" équations ont subi les calculs d'élimination.
La k° élimination est terminée.

0033 60 0058 0215 (0058) = oo (n - k + 1) oooo
0215 11 0072 0282 (0072) = oo oool oooo
0282 21 0058 0211 (0058)
0211 44 0200 0116

Les "n" éliminations n'ont pas été effectuées.

0200 (13) Nous commençons une nouvelle élimination.

La résolution du système linéaire et l'inversion de la matrice du système sont terminées.

0116 Nous effectuons l'instruction placée en 8001 lors de l'ENTREE dans le SOUS PROGRAMME de RESOLUTION de SYSTEMES LINEAIRES et d' INVERSION de MATRICE.

REMARQUES concernant le choix du premier pivot.

REMARQUE N° 1

```
0009    44 0007 0152        Test de fin du premier membre
                            d'une équation.
```

L'examen du premier membre de l'équation n'est pas terminé.

```
0007    nous reprenons le cours du programme général.
        ( en (11) page 118 )
```

L'examen du premier membre de l'équation est terminé.
Nous préparons le passage à l'équation suivante en supprimant l'étude des seconds membres.

```
0152    60 8001 0202        (8001) = 69 xxxx 0153
0202    10 0162 0252        (0162) = oo ( p ) oooo
                            (8003) = 69 xxxx 0153
                            en adresse facteur, nous avons
                            l'adresse de la mémoire
                            contenant le premier terme de
                            l'équation suivante.
0252    21 0064 0302        (0064)
0302    10 0040 0110        (0040) = - 48 0001 0096
0110    21 0014 0233        (0014) =   21 xxxx 0057
                            en adresse facteur, nous avons
                            l'adresse, diminuée de l'unité,
                            de la mémoire contenant le
                            premier terme de l'équation
                            suivante.
0233    nous reprenons le cours du programme général.
```

(page 121)

REMARQUE N° 2

```
0021    20 0096 0160        (0096) = 61 xxxx 0100
0160    65 0064 0210        (0064) = 69 xxxx 0153
0210    15 0260 8002        (0260) = - 02 0000 0056
8002    67 xxxx 0097
0097    nous reprenons le cours normal du programme
        général.
```

(page 119)

I, 3 SOUS-PROGRAMMES de RESOLUTION de SYSTEMES LINEAIRES en SIMPLE PRECISION et en DOUBLE PRECISION. Méthode de GAUSS. (Références G S P 1 et G D P 1)

I, 3-1 NOTICES d'UTILISATION

I, 3-1-1 SOUS-PROGRAMME en SIMPLE PRECISION

Objet : Etant donné la matrice et les vecteurs seconds membres d'un système linéaire, ce sous-programme calcule en virgule flottante *Flair* et en simple précision, les vecteurs solutions.

Conditions d'utilisation : Si «n» désigne l'ordre de la matrice du système linéaire, et «p» le nombre de vecteurs seconds membres, la limite d'utilisation de ce sous-programme est :

$$(n+1)\ (n+p)\ +\ n \leq 1462$$

Ainsi, avec p = 1, nous avons :

$$n_{MAX} = 36$$

La virgule flottante simulée étant du type *Flair*, on représentera :

1 par 51 1000 0000 +

Il faudra enregistrer l'A.V.F. (Arithmétique en Virgule Flottante simple précision) sur le tambour.

Utilisation : Le système à résoudre doit être rangé ligne par ligne dans des mémoires consécutives du tambour. Chaque ligne se composant des «n» coefficients du premier membre et des «p» coefficients des seconds membres, c'est-à-dire d'une équation complète du système.

Le calcul utilise $(2n + p - 1)$ mémoires auxiliaires pour les calculs et résultats intermédiaires. Ces $(2n + p - 1)$ mémoires doivent se trouver en séquence, le bloc qu'elles occupent étant situé immédiatement après celui occupé par les données.

Si nous désignons par «M» l'adresse de la première mémoire du bloc où est rangé le système (ligne par ligne), l'utilisateur devra placer en :

8003	00	0000	M	+
8002	00	n	$n + p$	+
8001	instruction de sortie			

Après les calculs, les mémoires qui contenaient les p séries de seconds membres contiennent les p vecteurs solutions correspondants. Nous pouvons très bien avoir $p > n$, la seule limite étant celle due à la formule du paragraphe précédent.

Exemple : Supposons : $n = 6$, $p = 2$ et $M = 0125$.

Les contenus de 8003 et 8002 à l'entrée, seront les suivants :

8003	00	0000	0125
8002	00	0006	0008

Les mémoires 0125 à 0172 seront réservées aux données (matrice et seconds membres). La 3° ligne du système sera enregistrée dans les mémoires 0141 à 0148.

Les mémoires 0173 à 0185 seront les mémoires auxiliaires pour les calculs et résultats partiels.

Les résultats occuperont pour :

- la première série de seconds membres, les mémoires : 0131, 0139, 0147, 0155, 0163, 0171,
- la deuxième série de seconds membres, les mémoires : 0132, 0140, 0148, 0156, 0164, 0172.

Forme du sous-programme : L'utilisateur dispose de deux paquets de cartes, dont les contenus devront être enregistrés sur le tambour :

1°/ *l'A V F* (Arithmétique en Virgule Flottante simple précision). Ce programme occupe les mémoires 1800 à 1999 sauf les zones entrée et sortie standard de la bande 1950 (1950 – 1960 et 1977 – 1986).

Il se présente sous forme de cartes «chargement 7 mots» (cartes roses) et est précédé d'un programme de chargement (cartes jaunes) qui utilise la bande 0000. Il faudra donc prévoir son chargement en temps utile dans le cas où l'on utiliserait la bande 0000 pour stocker les données ou à toute autre fin.

2°/ *Le SOUS-PROGRAMME* de Résolution de Systèmes Linéaires en Simple Précision. Il se présente sous forme numérique translatable «1 mot par carte» (cartes blanches).

Le facteur de translation Δ qui peut être quelconque mais inférieur ou égal à 1461, sera chargé en 1950.

La première instruction de ce sous-programme se trouve en Δ + 0000.

Encombrement : 200 mémoires pour l'A.V.F. (obligatoirement de 1800 à 1999, les zones entrée et sortie standard de la bande 1950 étant disponibles).

339 mémoires pour le sous-programme.

Arrêt programmé : Lorsque la matrice du système est singulière, nous avons l'arrêt :

01 0000 9999

Méthode utilisée : La méthode utilisée est la méthode de GAUSS avec choix du pivot maximal dans la matrice.

Temps de calcul : Avec p = 1, nous avons les temps de calcul ci-dessous :

n = 6	t = 28 secondes
n = 12	t = 2 minutes 39 secondes
n = 18	t = 7 minutes 56 secondes
n = 24	t = 17 minutes 40 secondes
n = 30	t = 33 minutes 09 secondes
n = 36	t = 56 minutes 50 secondes.

I, 3-1-2 SOUS-PROGRAMME en DOUBLE PRECISION

Objet : Etant donné, en simple ou en double précision, la matrice et les vecteurs seconds membres d'un système linéaire, ce sous-programme calcule en virgule flottante *Flair* et en double précision, les vecteurs solutions.

Conditions d'utilisation : Si «n» désigne l'ordre de la matrice du système linéaire, et «p» le nombre de vecteurs seconds membres, la limite d'utilisation de ce sous-programme est :

$$2(n+1)(n+p)+n \leqslant 1320$$

Ainsi avec p = 1, nous avons $n_{MAX} = 24$.

La virgule flottante simulée étant du type *Flair*, on représentera :

1 par $\underbrace{51\ 1000\ 0000}_{\text{I}} + \underbrace{00\ 0000\ 0000}_{\text{II}} +$

Dans tout ce qui suit, nous appellerons I : la partie gauche et II : la partie droite d'un nombre en virgule flottante double précision.

Il faudra enregistrer l'A.D.P. (Arithmétique en virgule flottante Double Précision) sur le tambour.

Utilisation : Le système à résoudre doit être rangé ligne par ligne dans des mémoires consécutives du tambour. Chaque ligne se composant des n coefficients du premier membre et des p coefficients des seconds membres, c'est-à-dire d'une équation complète du système. Le rangement du système en double précision se fera de la façon suivante :

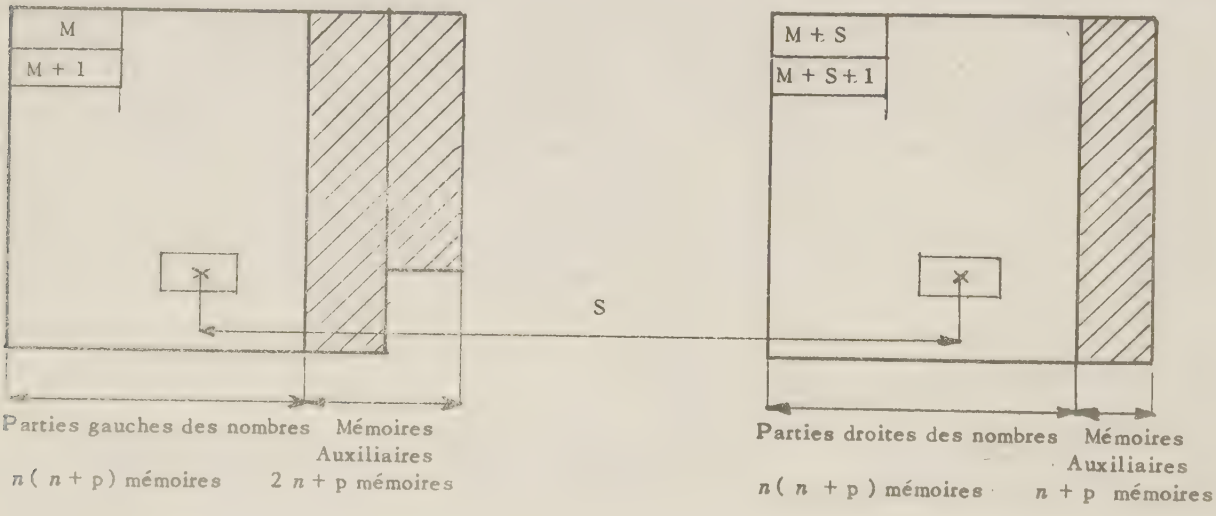
Un premier bloc de $(n + p)\, n$ mémoires dont la première a pour adresse «M» contiendra toutes les parties gauches des nombres en double précision.

Un second bloc de $(n + p)\, n$ mémoires dont la première a pour adresse «M + S» $(S > 0)$ contiendra toutes les parties droites des nombres en double précision.

Le calcul utilise $[2(n + p) + n]$ mémoires auxiliaires pour les calculs et résultats intermédiaires. Ces mémoires auxiliaires se répartissent en deux blocs disposés de la façon suivante :

- $(2n + p)$ mémoires en séquence, situées immédiatement à la suite du bloc de mémoires débutant en M et contenant les parties gauches des nombres en double précision,
- $(n + p)$ mémoires en séquence, situées immédiatement à la suite du bloc de mémoires débutant en M + S et contenant les parties droites des nombres en doubleprécision.

Nous aurons ainsi le schéma suivant :

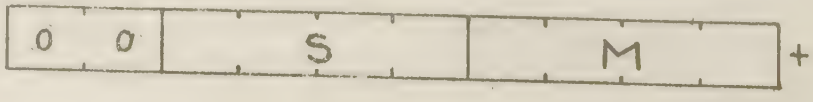


Remarque : Nous noterons qu'il faut $S \geqslant n\,(n + p) + 2n + p$.

D'autre part, dans le cas où les données sont en simple précision, il ne faut pas oublier de s'assurer que les mémoires devant contenir, au cours du calcul, les parties droites des nombres en double précision sont bien initialement soit à zéro + soit à zéro -.

Avec les notations déjà définies, pour entrer dans le sous-programme de résolution de systèmes linéaires en double précision, l'utilisateur devra placer en :

8003



8002 | 0 0 | 0 0 n | n+p | +

8001 une instruction de Sortie

Après les calculs, les mémoires qui contenaient les p séries de seconds membres contiennent les p vecteurs solutions correspondants. Nous pouvons très bien avoir $p > n$, la seule limite étant celle dûe à la formule du paragraphe précédent.

Exemple : Supposons : $n = 6$, $p = 2$, M = 0125 et S = 0102 . Les contenus de 8003 et 8002 à l'entrée, seront les suivants :

8003 : 00 0102 0125

8002 : 00 0006 0008

Les mémoires 0125 à 0172 contiennent les parties gauches des données ; les mémoires 0227 à 0274 contiennent les parties droites. Ainsi, la 3° ligne du système occupe les mémoires 0141 à 0148 et 0243 à 0250 . Les mémoires 0173 à 0185 et 0275 à 0282 seront les mémoires auxiliaires pour les calculs et résultats intermédiaires . Une fois les calculs terminés, les vecteurs solutions se trouvent à la place des vecteurs seconds membres :

- le 1° vecteur solution occupe les mémoires : 0131, 0139, 0147, 0155, 0163, 0171 pour les parties gauches des composantes en double précision et : 0233, 0241, 0249, 0257, 0265, 0273 pour les parties droites.
- le 2° vecteur solution occupe les mémoires : 0132, 0140, 0148, 0156, 0164, 0172 pour les parties gauches des composantes en double précision et : 0204, 0242, 0250, 0258, 0266, 0274 pour les parties droites.

Forme du sous-programme : l'utilisateur dispose de deux paquets de cartes, dont les contenus devront être enregistrés sur le tambour :

1°/ *l'A D P* (Arithmétique en virgule flottante Double Précision) . Ce programme occupe les mémoires 1700 à 1999, sauf les zones entrée et sortie standard de la bande 1950 (1950 à 1960 et 1977 à 1986) .

Il se présente sous forme de cartes «chargement 7 mots par carte» (cartes roses) et est précédé d'un programme de chargement (cartes jaunes) qui utilise la bande 0000 . Il faudra donc prévoir son chargement en temps utile dans le cas où l'on désire utiliser la bande 0000 : soit pour stocker les données, soit à toute autre fin.

2°/ *Le SOUS-PROGRAMME* de Résolution de Systèmes Linéaires en Double Précision . Il se présente sous forme numérique translatable «1 mot par carte» (cartes blanches) .

Le facteur de translation Δ, qui peut être quelconque mais inférieur ou égal à 1320, sera chargé en 1950 .

La première instruction de ce sous-programme se trouve en Δ + 0000 .

Encombrement : 300 mémoires pour l'A.D.P. (obligatoirement 1700 à 1999 ; les zones entrée et sortie standard de la bande 1950 étant disponibles) . 380 mémoires pour le sous-programme .

Arrêt programmé : lorsque la matrice du système est singulière, nous avons l'arrêt :

01 0000 9999

Méthode utilisée : La méthode utilisée est la méthode de GAUSS avec choix du pivot maximal dans toute la matrice .

Nous appellerons " n " l'ordre du système à résoudre, et " p " le nombre de vecteurs seconds membres.

Les données sont enregistrées sur le tambour à partir d'une mémoire d'adresse " M ". Le système est rangé équation par équation. Chaque équation comprend les premiers et les seconds membres.

Pour entrer dans le Sous Programme, il faut charger 8001, 8002, 8003 de la façon suivante:

(8001) = une instruction de sortie.
(8002) = oo (n)(n + p)
(8003) = oo oooo (M)

La première instruction du Sous Programme se trouve en 0000.

MEMOIRES de TRAVAIL

1°) à contenus constants

(0015) = 00 (M) 0000
(0022) = 61 [M + n − 1 + n(n + p)] 0100
en position facteur, nous avons la première adresse, diminuée de l'unité, de la série de "n + p" mémoires consécutives qui contiennent, pendant chaque élimination, l'équation pivot préalablement divisée par le pivot.
(0034) = 24 [M + n(n + p)] 0254
(0042) = 69 [M + n(n + p) + 1] 8002
Ces deux dernières instructions permettront de bloquer dans (n − k) mémoires consécutives à partir de l'adresse [M + n(n + p)], les rangs initiaux des inconnues restants à éliminer après la k° élimination.

(0066) = 21 [M + n(n + p) + n] 0255
Instruction pour le stockage des termes de l'équation pivot, divisés par le pivot, dans une série de mémoires de travail consécutives.
(0074) = 00 (n + p) 0000
(0078) = 60 [M + n(n + p)] 0178
Instruction utilisée lors des substitutions arrières, pour déterminer l'adresse de la mémoire où devra être placée la dernière racine calculée.
(0079) = 00 (n) 0000
(0088) = instruction de SORTIE du Sous-Programme de Résolution de Systèmes Linéaires.
(0124) = 69 [M + n(n + p)] 0303
Instruction utilisée pour l'appel du rang initial "l_i" de l'inconnue éliminée au cours de chaque élimination.
(0150) = 69 (M) 0103
Instruction pour l'appel des termes des équations convenables au cours de chaque élimination.
(0177) = 69 [M + n(n + p) + n + p − 1] 8002
Constante utilisée pour tester la fin du rangement de l'équation pivot à la fin de chaque élimination.
(0233) = 69 [M + n(n + p)] 0128
Constante utilisée, lors de chaque substitution arrière, pour la recherche du rang initial de la dernière inconnue calculée.

2°) à contenus variables.

(0001) = 00 (n − k + 1) 0000
avec $1 \leqslant k \leqslant n$, au cours des n éliminations.
= 00 (k − 1) 0000
avec $1 \leqslant k \leqslant n$, au cours de chacune des p substitutions arrières.
(0003) = successivement, au cours de chaque élimination:
− la valeur exacte du pivot.

- le multiplicateur relatif à chacune des (n-k) équations à traiter.

(0009) = Mot d'ORIENTATION
Son contenu sera:
44 0207 0202 pour le choix du premier pivot.
44 0207 0158 dans tous les autres cas.

(0014) = 00 0000 (n + p)
au tout début des calculs.

= 21 xxxx 0255
pour le stockage des termes de l'équation pivot divisés par le pivot, dans une série de mémoires de travail consécutives à partir de l'adresse $[M + n + n(n + p)]$.
Initialement, nous avons: (0014) = (0066)

= 21 xxxx 0257
pour le stockage des résultats de chaque élimination.
Au début de chacune d'elles, nous avons:
xxxx = M - 1

= 69 xxxx 0153
instruction d'appel des racines déjà calculées, au cours de la détermination, par substitution arrière, d'une nouvelle racine appartenant à la même série que les précédentes.
La valeur initiale de "xxxx" est toujours $[M+n+p-1]$ (avec p:variable). Nous passons à la valeur suivante en ajoutant (n + p) et ainsi de suite... (avec (n + p):constant)

(0017) = 60 AdrP 0050
pour la division de l'équation pivot par le pivot.
Nous avons en adresse facteur: "AdrP" qui représente l'adresse de la mémoire contenant le pivot.
Le contenu de la mémoire 0017 sera utilisé pour éviter la division du pivot par lui-même.

= 69 xxxx 0103
pour les calculs de la k° élimination.
Ce contenu servira à tester dans chaque équation:
- le terme de même rang que le pivot.
Nous aurons: "xxxx" = adresse de la mémoire contenant le terme de même rang que le pivot dans l'équation en cours de traitement.
- le dernier terme du premier membre.
Nous aurons: "xxxx" = adresse, augmentée de de l'unité, de la mémoire contenant le dernier terme du premier membre de l'équation en cours de traitement. Ce terme est de rang (n - k + 1).
- le dernier second membre
Nous aurons: "xxxx" = adresse, augmentée de l'unité, de la mémoire contenant le dernier second membre de l'équation en cours de traite - ment.

= résultats intermédiaires et valeur de chaque racine au cours des substitutions arrières.

(0021) = Mot d'ORIENTATION
Son contenu sera:
65 0064 0244 pour le choix du premier pivot.
20 0206 8001 dans tous les autres cas.

(0033) = Mot d'ORIENTATION
Son contenu sera:
69 0203 0106 pour le choix du premier pivot.
65 0064 0072 dans tous les autres cas.

(0039) = Au début de chaque élimination: la valeur absolue du pivot.
Ensuite cette mémoire est remise à zéro.
Au cours des calculs de chaque élimination, cette mémoire contient la valeur absolue du pivot pour la prochaine élimination.

(0051) = Au cours de chaque élimination:
- 00 0000 0000 pour toutes les équations de rang inférieur à k_p.
- 00 (n + p) 0000 pour toutes les équations de rang supérieur à k_p.

(0064) = 60 xxxx 0050
pour l'appel des termes de l'équation pivot afin de les diviser par le pivot.
Initialement, nous avons: "xxxx" = l'adresse de la mémoire contenant le premier terme de l'équation

pivot.

= 69 xxxx 0103

pour l'appel des termes des (n − k + 1) premières équations au cours de la k° élimination.

Nous avons initialement, xxxx = M .

= 61 xxxx 0014

pour l'appel de tous les termes du premier membre de chaque équation pivot au cours de chaque substitution arrière.

Initialement, "xxxx" = $\left[M + (k-1)(n+p)\right]$ chaque fois que nous calculons la k° racine de l'une des "p" séries à déterminer.

(0068) = 00 (l_p − 1) 0000

avec "l_p" représentant le rang occupé par l'inconnue à éliminer au cours de la k° élimination.

Remarque

Dans le cas du choix du premier pivot, nous avons:

(0068) = 00 (n) 0000

(0084) = 60 xxxx 0178

Instruction d'appel des rangs initiaux des inconnues déjà calculées au cours d'une même substitution arrière, pour les comparer à celui de la dernière racine obtenue.

Initialement: (0084) = (0078).

(0095) = comptage des équations au cours de chaque élimination.

Ce compteur est remis à zéro au début de chaque élimination.

= CO xxxx 0000

pendant la détermination de l'adresse de la mémoire où doit être placée la dernière racine calculée au cours d'une substitution arrière.

Au début de chaque détermination, nous faisons: "xxxx" = (M + n + p − 1) ; à la fin nous avons "xxxx" égal à l'adresse cherchée.

(0108) = 00 (p) 0000

Quantité constante pendant toutes les éliminations.

Quantité variable lors des substitutions arrières; nous passons d'une substitution à la suivante en changeant "p" en (p − 1).

(0116) = Mot d'ORIENTATION associé aux termes d'une même équation.

Son contenu sera successivement au cours du traitement de chaque équation:

CO 0088 0000

00 0089 0000

00 0090 0000

(0149) = 00 (k_p − 1) 0000

Pendant toute la k° élimination, "k_p" représente le rang actuel de l'équation pivot.

Remarque

Au début du premier choix de pivot, nous avons:

(0149) = 00 (n) 0000

(0166) = 61 xxxx 0014

Cette quantité est utilisée pour tester la fin de la détermination de la valeur de chaque racine pendant tous les calculs des substitutions arrières.

Ainsi lorsque nous calculons les k° racines, nous avons: "xxxx" = $\left[M + k - 1 + (k-1)(n+p)\right]$. Il s'agit de l'adresse, augmentée de l'unité, du dernier terme du premier membre de la (n − k + 1)° équation pivot.

(0206) = 00 (l_i − 1) 0000

au début de chaque élimination, avec "l_i" représentant le rang initial de l'inconnue en cours d'élimination.

= 61 xxxx 0100

Instruction d'appel des coefficients de l'équation pivot au cours de chaque élimination et pour le traitement de chaque équation. Il s'agit de l'équation pivot divisée par le pivot et stockée dans une série de mémoires de travail consécutives.

Au début du traitement de chaque équation, nous avons: (0206) = (0022).

= 00 (l_i - 1) 0000
au cours des substitutions arrières, avec "l_i" représentant le rang initial de la dernière inconnue calculée.

(0222) = 00 (M + n + p - 1) 0000
Quantité utilisée au cours des substitutions arrières. La valeur de "p" est variable et diminue d'une unité chaque fois que l'on passe d'une substi - tution arrière à la suivante.

(0300) = 21 AdrP 0257
Au début de chaque élimination, "AdrP" représente l'adresse de la mémoire contenant la valeur exacte du pivot pour l'élimination en cours.
Ensuite, "AdrP" représente l'adresse de la mémoire contenant la valeur exacte du pivot pour l'élimination suivante.

INSTRUCTIONS

Initialisation.

```
0000   24 0088 0041
0041   69 8003 0198
0198   23 0014 0334      (0014) = oo oooo (n + p)
0334   69 0001 0135      (0001) = oo xxxx oooo
0135   22 0079 0185      (0079) = oo ( n ) oooo
0185   24 0149 0052
0052   24 0068 0235
0235   24 0001 0285
0285   35 0004 0195
0195   22 0074 0335      (0074) = oo (n + p) oooo
0335   16 0079 0086
0086   22 0108 0136      (0108) = oo ( p ) oooo
0136   65 8003 0143
0143   24 0016 0186      (0016) = oo ( M ) oooo
0186   69 0150 0236      (0150) = 69 xxxx 0103
0236   22 0150 0286      (0150) = 69 ( M ) 0103
0286   10 0014 0336      (0014) = oo oooo (n + p)
0336   19 0079 0101      (0079) = oo ( n ) oooo
                         (8003) = oo ( M ) oooo
                         (8002) = oo [n(n + p)] oooo
0101   15 8003 0337      (8002) = oo [M+n(n+p)] oooo
                         en position facteur, la première adresse de la série de " n " mémoires utilisées pour la mise en ordre des résultats.
0337   69 0233 0138      (0233) = 69 xxxx 0128
0138   22 0233 0188      (0233) = 69 [M+n(n+p)] 0128
0188   69 0078 0238      (0078) = 60 xxxx 0178
0238   22 0078 0288      (0078) = 60 [M+n(n+p)] 0178
0288   15 0091 0245      (0091) = 69 0001 8002
0245   20 0042 0295      (0042) = 69 [M+n(n+p)+1] 8002
0295   16 0248 0338      (0248) = 00 0001 7699
0338   20 0124 0289      (0124) = 69 [M+n(n+p)] 0303
0289   69 0034 0090      (0034) = 24 xxxx 0254
0090   22 0034 0140      (0034) = 24 [M+n(n+p)] 0254
```

Chargement des mémoires utilisées pour la mise en ordre des résultats des " p " substitutions arrières. Au début des calculs, la mémoire d'adresse [M+n(n+p)+j-1] est associée à la j° inconnue et sera chargée avec oo (j - 1) oooo.

```
       0140   65 8002 0151
       0151   15 0190 8002      (0190) = - 48 0000 0063
(1)    8002   21 xxxx 0240
       0240   10 0007 0290      (0007) = 00 0001 0000
       0290   15 8001 0201
       0201   11 0079 0141      (0079) = 00 ( n ) 0000
       0141   44 0046 0096
```

Le chargement des " n " mémoires n'est pas terminé.

```
       0046   10 8001 8002  (1)
```

Le chargement des " n " mémoires est terminé.

0096	15 0251 0191	(0251) = 00 0000 0015
0191	20 0066 0241	(0066) = 21 [M+n(n+p)+n] 0255

en position facteur, la première adresse d'une série de (n+p) mémoires de travail utilisées au stockage de l'équation pivot au cours de chaque élimination.

0241	15 0044 0301	(0044) = 39 9998 9845
0301	20 0022 0291	(0022) = 61 [M+n(n+p)+n-1] 0100
0291	15 0108 0292	(0108) = 00 (p) 0000
0292	69 0177 0193	(0177) = 69 xxxx 8002
0193	22 0177 0243	(0177) = 69 [M+n(n+p)+n+p-1] 8002

Préparation de la recherche du premier pivot.

0243	69 0146 0102	(0146) = 44 0207 0202
0102	24 0009 0293	
0293	69 0196 0152	(0196) = 65 0064 0244
0152	24 0021 0094	
0094	69 0250 0144	(0250) = 69 0203 0106
0144	24 0033 0118 (2)	

Maintenant la recherche du premier pivot peut s'effectuer. Cette recherche présente un aspect particulier. En effet, et celà dès la première élimination, nous ferons le choix du pivot pour la (k+1)° élimination en même temps que nous ferons les calculs de la k° élimination.
Nous allons par conséquent exposer la suite du programme dans le cas général en mettant en évidence les embranchements particuliers à la recherche du premier pivot.

K° ELIMINATION.

Au début de cette élimination, nous avons (n-k+1) équations formées chacune de (n-k+1+p) termes. Les termes d'une même équation sont enregistrés dans des mémoires consécutives. Les premiers termes des (n-k+1) équations sont dans les mémoires d'adresses: M , M+n(n+p) , M+2(n+p) , M+3(n+p) , ... , M+(n-k)(n+p).

A la suite de ces (n-k+1) équations, nous trouvons les (k-1) équations pivots des (k-1) premières éliminations. L'équation pivot de la (k-1)° élimination comprend (n-k+1+p) termes placés dans des mémoires consécutives à partir de l'adresse: M+(n-k+1)(n+p). L'équation pivot de la (k-2)° élimination se compose de (n-k+2+p) termes placés dans des mémoires consécutives à partir de l'adresse: M+(n-k+2)(n+p). Et ainsi de suite..., jusqu'à l'équation pivot de la première élimination. Cette dernière comprend (n-1+p) termes conservés dans des mémoires consécutives à partir de l'adresse: M+(n-1)(n+p)

Le pivot de la k° élimination, qui a été choisi en même temps que s'effectuaient les calculs de l'élimination précédente (mis à part le cas du premier pivot) est caractérisé par:

- sa valeur absolue: contenu de la mémoire 0039.
- son adresse: en position facteur du contenu de la mémoire 0014.

A l'aide de ce dernier renseignement, nous calculerons:

- " k_p ": le numéro de l'équation dans laquelle a été trouvé le pivot.
- " l_p ": le numéro de l'inconnue dont il est le coefficient.

Notons que nous aurons toujours k_p et l_p inférieurs ou égaux à (n-k+1).

A ce moment, les " n " mémoires consécutives, initialement associées aux " n " inconnues, sont scindées en deux parties. La première est composée des

(n-k+1) premières mémoires associées aux inconnues restant à éliminer. Ainsi le contenu de la mémoire d'adresse M+n(n+p) $+l_p-1$ nous donne l'ordre initial de la l_p° inconnue actuelle. Nous le désignerons par " l_i ". La seconde partie est composée par les (k-1) dernières mémoires associées aux équations pivots des (k-1) premières éliminations. Le contenu de chacune de ces mémoires indique le rang de l'inconnue que permettra de calculer l'équation correspondante, lors des substitutions arrières.

Avant de commencer la k° élimination, nous mettrons à jour les " n " mémoires précédentes. Pour celà, nous bloquerons les renseignements relatifs aux inconnues restant à éliminer dans les (n-k) premières mémoires. Dans la (n-k+1)° mémoire, nous placerons le renseignement relatif à l'équation pivot de la k° élimination (c'est à dire l'ancien contenu de la mémoire d'adresse $M+n(n+p)+l_p-1$). Les contenus des autres mémoires restent inchangés.

Ensuite nous diviserons l'équation pivot par le pivot et transfèrerons les résultats dans des mémoires de travail consécutives, à partir de l'adresse M+n(n+p)+n. Puis la k° élimination terminée, nous transfèrerons cette équation pivot à la suite des résultats de cette élimination: en position de (n-k+1)° équation. Les équations pivots des éliminations précédentes ne sont pas affectées par ces calculs. La situation sera alors comparable à ce qu'elle était au début de la k° élimination, nous pourrons passer à l'élimination suivante.

Restauration des aiguillages.

(3)	0250	69 0203 0106		(0203) = 44 0207 0158
	0106	24 0009 0012		
	0012	69 0015 0018		(0015) = 20 0206 8001
	0018	24 0021 0024		
	0024	69 0027 0030		(0027) = 65 0064 0072
	0030	24 0033 0036	(4)	
(4)	0036	60 0039 0043		(0039) = valeur absolue du pivot.
	0043	44 0047 0048		

Le pivot est nul.

0048	01 0000 9999	La matrice du système est singulière, on ne peut pas poursuivre les calculs.

Le pivot n'est pas nul.

0047	65 0300 0055	(0300) = 21 AdrP 0257
		AdrP: adresse du pivot.
0055	15 0008 8002	(0008) = 47 9999 9996
8002	69 AdrP 0253	
0253	24 0003 0156	(0003) = valeur exacte du pivot.
0156	16 0059 0013	(0059) = 69 0000 0253
0013	16 0016 0071	(0016) = 00 (M) 0000
		(8002) = 00 (AdrP-M) 0000
0071	14 0074 0035	(0074) = 00 (n + p) 0000
		(8002) = 00 0000 (k_p-1)
		(8003) = 00 (l_p-1) 0000
0035	35 0004 0045	
0045	20 0149 0002	(0149) = 00 (k_p-1) 0000
0002	30 0004 0063	
0063	21 0068 0121	(0068) = 00 (l_p-1) 0000

Recherche de l'ordre initial de l'inconnue à éliminer au cours de la k° élimination.

0121	10 0124 8003	(0124) = 69 [M+n(n+p)] 0303

	8003	69 xxxx 0303	Avec "xxxx" = M+n(n+p)+l_p-1
	0303	24 0206 0109	(0206) = 00 (l_i-1) 0000

Blocage, à partir de la mémoire d'adresse M+n(n+p), des (n-k) mémoires associées aux inconnues restant à éliminer après la k° élimination.

Au cours de cette élimination, l'inconnue d'ordre l_p disparait. Par conséquent, tous les coefficients des inconnues d'ordres supérieurs à l_p, qui restent à éliminer, vont gagner une position. Il faut donc que les mémoires associées à ces inconnues subissent un déplacement analogue pour que leurs contenus conservent toute leur signification. En conséquence, nous commencerons à travailler sur la mémoire associée à la (l_p+1)° inconnue. Son adresse est égale à M+n(n+p)+l_p.

	0109	60 0068 0023	(0068) = 00 (l_p-1) 0000
	0023	15 8001 0031	
	0031	15 0034 0089	(0034) = 24 [M+n(n+p)] 0254
(5)	0089	10 0042 8003	(0042) = 69 [M+n(n+p)+1] 8002
			(8003) = 69 [M+n(n+p)+l_p] 8002
			(8002) = 24 [M+l_p-1 / +n(n+p)] 0254
	8003	69 xxxx 8002	
	8002	24 xxxx 0254	
	0254	10 0007 0211	(0007) = 00 0001 0000
	0211	15 8001 0019	
	0019	11 0042 0097	(0042)
	0097	11 0001 0105	(0001) = 00 (n-k+1) 0000
	0105	44 0159 0010	Test de fin de blocage des (n-k) mémoires associées aux inconnues restant à éliminer.

Le blocage n'est pas terminé.

	0159	10 8001 0089	(5)

Le blocage est terminé.

Chargement de la mémoire associée à l'équation pivot de la k° élimination.

	0010	15 0113 0067	(0113) = - 00 0000 9950
			(8002) = 24 [M+n-k / +n(n+p)] 0304
	0067	69 0206 8002	(0206) = 00 (l_i-1) 0000
	8002	24 xxxx 0304	

Transfert de l'équation pivot dans des mémoires de travail.

Nous allons diviser les (n-k+1+p) termes de l'équation pivot, situés dans des mémoires consécutives à partir de l'adresse M+(k_p-1)(n+p), par le pivot d'adresse AdrP = M+(k_p-1)(n+p)+l_p-1.

La division du pivot par lui-même donne l'unité. Nous n'effectuerons pas ce calcul, et nous ne mettrons pas le résultat correspondant en mémoire.

Les (n-k+p) autres résultats seront mis dans des mémoires consécutives de travail à partir de l'adresse M+n(n+p)+n.

	0304	60 0300 0155	(0300) = 21 AdrP 0257
	0155	10 0058 0163	(0058) = 38 9999 9793
	0163	15 0066 0171	(0066) = 21 [M+n(n+p)+n] 0255
	0171	21 0017 0070	(0017) = 60 AdrP 0050
	0070	11 0068 0073	(0068) = 00 (l_p-1) 0000
(6)	0073	21 0064 0117	(0064) = 60 (AdrP-l_p+1) 0050
			AdrP-l_p+1 = M+(k_p-1)(n+p)
	0117	20 0014 0167	(0014) = 21 [M+n(n+p)+n] 0255
	0167	11 0017 0221	(0017)
	0221	44 0025 0026	

Le prochain terme à diviser est le pivot. Nous passons à la division suivante sans effectuer d'opération.

	0026	60 0007 0069	(0007) = 00 0001 0000
	0069	10 0064 0119	(0064)
	0119	15 0014 0073	(6) (0014)

Le prochain terme à diviser n'est pas le pivot, nous effectuons l'opération.

0025	10 0068 0123	(0068) = 00 (l_p-1) 0000
0123	11 0001 0205	(0001) = 00 (n-k+1) 0000
0205	11 0108 0213	(0108) = 00 (p) 0000
0213	44 0064 0118	Test de fin des divisions.

Toutes les divisions n'ont pas été effectuées.

0064 60 xxxx 0050

Séquence d'Entrée dans l'A.V.F. pour effectuer une division.

0050	69 0003 0006	(0003) = diviseur.
0006	24 1801 0004	
0004	69 1900 0053	VDV
0053	24 1808 0011	
0011	69 0014 1800	(0014) = instruction de sortie.

1800

Sous-Programme
A. V. F.

1803 (1803) = (0014)

Mise en mémoire du résultat.

1803 21 xxxx 0255

Préparation de la division suivante.

0255	60 0007 0261	(0007) = 00 0001 0000
0261	15 8001 0069	
0069	10 0064 0119	(0064)
0119	15 0014 0073	(6)

Toutes les divisions sont terminées, nous commencons les calculs d'élimination.
Pour ces calculs, nous devrons répondre aux impératifs suivants:

1°) ne faire aucun calcul avec l'équation pivot.
Il s'agit de la k_p° équation.

2°) limiter les calculs et le choix de pivot pour l'élimination suivante aux (n-k+1) premières équations.

3°) détecter la fin de l'élimination.

Le premier impératif fera l'objet d'un test situé dans la mémoire 0306.

(0306) = 44 0259 0033

Les deux suivants seront réalisés à l'aide du test placé dans la mémoire 0057.

(0057) = 44 0311 0062

4°) à l'intérieur de chaque équation, repérer le terme de même rang que le pivot.
Il s'agit du l_p° terme.

5°) à l'intérieur de chaque équation, tester la fin du premier membre ((n-k+1) termes).

6°) à l'intérieur de chaque équation, détecter la fin des seconds membres; c'est à dire de l'équation.

Ces trois derniers impératifs feront l'objet d'un test situé dans la mémoire de travail 0009

(0009) = 44 0207 0158

et l'aiguillage sera du type 88, 89, 90 (contenus successifs de la mémoire 0116).

Signalons aussi les contenus de diverses mémoires de travail au début de ces calculs:

1°) Dans le cas général.

(0001) = 00 (n-k+1) 0000
(0009) = 44 0207 0158
(0021) = 20 0206 8001
(0033) = 65 0064 0072
(0068) = 00 (l_p-1) 0000
(0149) = 00 (k_p-1) 0000

2°) Dans le cas du choix du premier pivot.

(0001) = 00 (n) 0000
(0009) = 44 0207 0202
(0021) = 65 0064 0244
(0033) = 69 0203 0106
(0068) = 00 (n) 0000
(0149) = 00 (n) 0000

Enfin, sous forme de "REMARQUES" nous indiquerons les embranchements particuliers à la recherche du premier pivot.

(2)	0118	65 0150 0305	(0150) = 69 (M) 0103 REMARQUE C'est par cette instruction que commence le choix du premier pivot.
	0305	24 0064 0217	(0064)
	0217	21 0039 0092	(0039) = 00 0000 0000 Remise à zéro de la mémoire contenant la valeur maximum du pivot, en vue du prochain choix de pivot.
	0092	24 0095 0098	
	0098	24 0051 0256	
	0256	15 0209 0263	(0209) = - 48 0000 9846 (8002) = 21 (M) 0257
	0263	20 0014 0267	(0014)
(7)	0267	11 0149 0306	
	0306	44 0259 0033	Test relatif à l'équation pivot.

L'équation que nous devons traiter n'est pas l'équation pivot.

0259	10 8001 0317	
0317	11 0001 0057	
0057	44 0311 0062	Test relatif à la fin de l'élimination en cours.

Les (n-k+1) premières équations n'ont pas été toutes traitées.

0311	65 0064 0169	(0064)
0169	15 0068 0173	
0173	20 0017 0120	(0017) = 69 xxxx 0103 en position facteur, nous avons l'adresse du terme de même rang que le pivot dans l'équation traitée.
0120	15 0223 8002	(0223) = 00 0000 0004
8002	69 xxxx 0107	
0107	24 0003 0157	(0003) = multiplicateur pour les calculs d'élimination relatifs à une équation.
0157	69 0060 0313	(0060) = 00 0088 0000
0313	24 0116 0219	(0116)
0219	69 0022 0075	(0022) = 61 [M+n(n+p)+n-1] 0100
0075	24 0206 0309	(0206)
(8) 0309	60 0064 0269	(0064)
0269	11 0017 0009	0009 Mémoire de travail utilisée pour l'orientation du programme. Pour l'étude du choix du premier pivot, se reporter à la REMARQUE N°1. Nous poursuivons l'étude du programme dans le cas général.
0009	44 0207 0158	Test relatif aux termes d'une même équation.

Le terme sur lequel nous devons opérer n'est pas:

- de même rang que le pivot.
- de rang (n-k+2). Fin du traitement du premier membre.
- le dernier terme de l'équation complète.

0207	60 0014 0319	(0014)
0319	15 0206 0112	(0206)

```
0112    10 0007 0162        (0007) = 00 0001 0000
0162    15 8001 0271
0271    21 0014 0021
```

0021 Mémoire de travail utilisée pour l'orientation du programme.

Pour l'étude du choix du premier pivot, se reporter à la REMARQUE N°2.

Nous poursuivons l'étude du programme dans le cas général.

```
0021    20 0206 8001
8001    61 xxxx 0100
```

Séquence d'Entrée dans l'A.V.F. pour éxécuter une multiplication.

```
0100    69 0003 0056        (0003) = multiplicateur
0056    24 1801 0054
0054    69 1996 0049        VMU
0049    24 1808 0061
0061    69 0064 1800        (0064) = instruction de Sortie
```

```
1800
Sous-Programme
A. V. F.
                1803        (1803) = (0064)
```

Séquence d'Entrée dans l'A.V.F. pour effectuer une addition.

```
1803    69 xxxx 0103
0103    24 1801 0104
0104    69 1902 0053        VAA
0053    24 1808 0011
0011    69 0014 1800        (0014) = instruction de Sortie
```

```
1800
Sous-Programme
A. V. F.
                1803        (1803) = (0014)
```

Mise en mémoire du résultat.

```
1803    21 xxxx 0257
0257    69 0116 0170        (0116) = 00 008/9x 0000
0170    96 0273 0125
```

Nous travaillons sur le premier membre d'une équation, le choix de pivot pour l'élimination suivante doit avoir lieu.

```
0273    67 8003 0081        (8002) = valeur absolue du
                            dernier coefficient calculé.
0081    16 0039 0093        (0039) = valeur absolue du
                            pivot actuellement retenu.

0093    46 0125 0147
```

Le dernier résultat obtenu est le maximum en valeur absolue et doit être retenu comme pivot.

```
0147    15 8001 0307
0307    20 0039 0142        (0039) = nouvelle valeur
                            absolue du pivot actuellement
                            retenu.
0142    69 0014 0168        (0014)
0168    24 0300 0125        (0300) = en position facteur,
                            l'adresse de la mémoire qui
                            contient la valeur exacte du
                            terme retenu comme pivot.
```

Le choix du pivot est terminé.
Ou il n'y a pas eu de choix de pivot.
Dans les deux cas, préparation du calcul suivant.

```
(9)     0125    60 0064 0220        (0064)
        0220    10 0007 0212        (0007) = 00 0001 0000
        0212    21 0064 0309    (8)
```

Réponse négative au test 44 0207 0158.
La série de calculs en cours sur une partie d'équation est terminée.

```
0158    60 8001 0218        (8001) = 69 xxxx 0103
0218    15 0007 0262        (0007) = 00 0001 0000
0262    15 0116 0321        (0116) = 00 008/98/9 0000
0321    96 0174 0076
0174    95 0077 0029
```

Tous les termes de l'équation, de rang inférieur à celui du pivot, ont subi les calculs d'élimination.
Préparation d'une nouvelle série de calculs qui portera sur tous les autres termes du premier membre.

```
0077    11 0068 0323        (0068) = 00 (l_p-1) 0000
0323    10 0001 0208        (0001) = 00 (n-k+1) 0000
0208    21 0017 0270        (0017) = en position facteur:
                            l'adresse, augmentée de l'unité,
                            du dernier terme du premier
                            membre de l'équation traitée.
0270    20 0116 0125  (9)   (0116) = 00 0089 0000
```

Remarque: En 0125, nous modifions l'instruction d'appel des coefficients de l'équation, de façon à ne faire aucun calcul avec le terme de même rang que le pivot.
Par contre, nous ne modifions pas l'instruction de mise en mémoire des résultats.

Tout le premier membre de l'équation a subi les calculs d'élimination.
Préparation, pour cette équation, de la fin des calculs qui porteront sur les "p" seconds membres.
Nous aurons un 9 en position 6 du contenu de 0116, d'où la suppression du choix du pivot.

```
0029    10 0108 0214        (0108) = 00 ( p ) 0000
0214    21 0017 0320        (0017) = en position facteur:
                            l'adresse, augmentée de l'unité,
                            du dernier second membre de
                            l'équation traitée.
0320    20 0116 0309  (8)   (0116) = 00 0090 0000
```

L'équation entière a subi les calculs d'élimination.
Nous allons préparer le passage à l'équation suivante.

```
0076    10 0079 0083        (0079) = 00 ( n ) 0000
0083    11 0001 0258        (0001) = 00 (n-k+1) 0000
0258    21 0064 0268        (0064) = 69 xxxx 0103
                            l'adresse facteur est celle
                            du premier terme de l'équation
                            suivante.
0268    10 0209 0264        (0209) = - 48 0000 9846
0264    11 0051 0308        (0051) = voir la remarque
                                     ci-dessous.
0308    21 0014 0318        (0014) = instruction pour la
                            mise en mémoire des résultats
                            suivants.
(10) 0318  60 0095 0199     (0095) = comptage des équations.
0199    10 0007 0312        (0007) = 00 0001 0000
0312    21 0095 0267  (7)
```

Remarque:
- (0051) = 00 0000 0000, pour toutes les équations de rang inférieur à k_p (équation pivot).
  La nouvelle équation obtenue par élimination sera mise à la place de l'équation dont elle est issue.
- (0051) = 00 (n+p) 0000, pour toutes les équations de rang supérieur à k_p.
  La nouvelle équation obtenue par élimination sera mise à la place de l'équation précédant celle dont elle est issue.

Réponse négative au test: 44 0259 0033

1°) Cas du choix du premier pivot.
Le choix du premier pivot est terminé, nous sommes prêts à commencer la première élimination.

```
0033    69 0203 0106
```

(voir page 133)

2°) Cas général.

L'équation que nous sommes prêts à traiter est la k_p° équation. Elle est "équation pivot" pour la k° élimination et ne doit pas subir de calcul. Nous passons à l'équation suivante.

```
0033    65 0064 0072          (0064) = 69 xxxx 0103
0072    15 0074 0129          (0074) = 00 (n+p) 0000
0129    24 0051 0110          (0051) = 00 (n+p) 0000
0110    20 0064 0318  (10)    (0064)
```

L'équation que nous sommes prêts à traiter est la (n-k+2)° équation. La k° élimination est terminée.
Avant de passer à l'élimination suivante, nous allons ranger l'équation pivot en position de (n-k+1)° équation, à la suite des résultats de la k° élimination.
Pour celà, nous allons translater les contenus des (n-k+p) mémoires de travail consécutives à partir de l'adresse [M+n(n+p)+n] dans des mémoires consécutives à partir de l'adresse [M+(n-k)(n+p)].

```
0062    60 0022 0127          (0022) = 61 [M+n(n+p)+n-1] 0100
0127    15 0014 0122          (0014) = 21 zzzz 0257
                              zzzz = M+(n-k)(n+p)-1
0122    10 0175 0179          (0175) = 08 0000 7902
0179    15 0032 0037          (0032) = 02 9999 9780
                              (8002) = 24 zzzz 0037
                              (8003) = 69 [M+n(n+p)+n-1] 8002
(11) 0037    10 0007 0314     (0007) = 00 0001 0000
     0314    15 8001 0224
     0224    11 0177 0131     (0177) = 69 [M+n(n+p) / +n+p-1] 8002
     0131    11 0001 0160     (0001) = 00 (n-k+1) 0000
     0160    44 0065 0115
```

La translation n'est pas terminée.

```
0065    10 8001 0274
0274    10 0177 8003          (0177)
8003    69 xxxx 8002
8002    24 xxxx 0037  (11)
```

La translation est terminée.
Toutes les inconnues ont-elles été éliminées?

```
0115    60 0001 0210          (0001) = 00 (n-k+1) 0000
0210    11 0007 0165          (0007) = 00 0001 0000
0165    21 0001 0260          (0001)
0260    44 0036 0215
```

Les "n" éliminations n'ont pas été effectuées.

```
0036    Nous commencons une nouvelle élimination.
        Voir en (4).
```

REMARQUES concernant le choix du premier pivot.

REMARQUE N°1

```
0009    44 0207 0202    Test de fin du premier membre
                        d'une équation.
```

L'examen du premier membre de l'équation n'est pas terminé.

```
0207    nous reprenons le cours du programme général.
        (voir page 136)
```

L'examen du premier membre de l'équation est terminé.
Nous préparons le passage à l'équation suivante en supprimant l'étude des seconds membres.

```
0202    60 8001 0194          (8001) = 69 xxxx 0103
                              en position facteur, nous avons
                              l'adresse, augmentée de l'unité,
                              du dernier terme du premier
                              membre de l'équation que nous
                              venons de traiter.
0194    10 0108 0258          (0108) = 00 ( p ) 0000
                              (8003) = 69 xxxx 0103
                              en position facteur, nous avons
```

l'adresse de la mémoire
contenant le premier terme de
l'équation suivante.

0258 nous reprenons le cours du programme général.

REMARQUE N°2

0021 65 0064 0244 (0064) = 69 xxxx 0103
en position facteur, nous avons
l'adresse du prochain terme à
examiner.

0244 15 0297 8002 (0297) = - 02 0000 0022

8002 67 xxxx 0081

0081 nous reprenons le cours normal du programme général
après avoir évité tout calcul sur les données.

Les "n" éliminations ont été effectuées.

0215 nous sommes prêts à commencer les
SUBSTITUTIONS ARRIERES.

SUBSTITUTIONS ARRIERES

Maintenant que les "n" éliminations sont effectuées, nous disposons de "n" équations pivots occupant la place des équations initiales. La première comprend (n-1+p) termes et occupe la place de la n° équation initiale. La seconde comprend (n-2+p) termes et occupe la place de la (n-1)° équation initiale. Et ainsi de suite ..... jusqu'à la n° équation pivot qui comprend p termes et qui occupe la place de la première équation donnée.

Associées à ces "n" équations pivots nous avons "n" mémoires consécutives à partir de l'adresse [M+n(n+p)] . Chacune d'elles contient, en position facteur, le rang initial moins un de l'inconnue que fournira l'équation pivot correspondante.

Les "p" substitutions arrières, correspondant aux "p" séries de seconds membres, seront effectuées indépendamment les unes des autres. Nous commencerons par la dernière (correspondant à la p° série de seconds membres) de façon à conserver les équations pivots intactes et à stocker les racines à la place initialement occupée par la série correspondante de seconds membres.

Au début des substitutions arrières, le contenu de la mémoire 0001 est nul.

Préparation d'une nouvelle substitution arrière.

(12)	0215	60 0016 0172	(0016) = 00 (M) 0000
	0172	15 8001 0181	
	0181	10 0079 0133	(0079) = 00 (n) 0000
	0133	10 0108 0265	(0108) = 00 (p) 0000
	0265	11 0007 0315	(0007) = 00 0001 0000
	0315	21 0222 0225	(0222) = 00 (M+n+p−1) 0000 en position facteur, nous avons la première adresse de la série de "n" mémoires où seront stockées les racines que nous nous apprêtons à calculer. Nous passerons de l'adresse d'une mémoire à celle de la suivante en ajoutant (n+p).
	0225	15 0028 0183	(0028) = 61 0000 0014 (8002) = 61 (M) 0014 en position facteur, nous avons l'adresse de la première des mémoires contenant la n° équation pivot. Nous passerons d'une adresse à la suivante en ajoutant l'unité.
	0183	Voir en (13)	

Nous sommes prêts à commencer le calcul de la première racine: k = 1. Le contenu de la mémoire 0001 est nul.

Calcul de la k° racine appartenant à la p° série de racines.

A ce moment les (k−1) mémoires d'adresses:

$M+n+p-1$
$M+2(n+p)-1$
$M+3(n+p)-1$
.....
$M+(k-1)(n+p)-1$

contiennent les racines déjà calculées et qui ont été convenablement classées (par ordres initiaux croissants) de la façon que nous verrons plus loin.

Les contenus de certaines mémoires de travail sont alors les suivants:

(0001) = 00 (k − 1) 0000
(0064) = 61 $\left[M+(k-1)(n+p)\right]$ 0014
en position facteur, nous avons l'adresse du premier terme de la (n−k+1)° équation pivot.
(0078) = 60 $\left[M+n(n+p)\right]$ 0178
(0222) = 00 (M+n+p−1) 0000

La détermination de la k° racine va se décomposer de la façon suivante:

1° PARTIE : Calcul de la valeur de la k° racine.

Pour cela, nous considérons la (n−k+1)° équation pivot qui comprend (k−1+p) termes stockés dans des mémoires consécutives à partir de l'adresse $\left[M+(k-1)(n+p)\right]$. Etant donné le mode opératoire des éliminations précédemment effectuées, les (k−1) termes du premier membre sont classés, comme les inconnues déjà calculées, suivants leurs rangs initiaux croissants. Nous pouvons donc considérer le p° second membre de cette équation et lui soustraire la somme des produits des termes du premier membre par les racines correspondantes. Le résultat obtenu sera la racine cherchée. En effet, le coefficient de cette inconnue dans l'équation pivot est égal à l'unité.

Nous allons commencer le calcul de la k° racine appartenant à la p° série de racines. La première instruction se trouve dans la mémoire 0183 et nous avons:

(8003) = 00 (M+n+p-1) 0000 pour k = 1.
00 0000 0000 pour toutes les autres valeurs de k.

(8002) = 61 $\left[M+(k-1)(n+p)\right]$ 0014
en position facteur, nous avons l'adresse de la mémoire contenant le premier terme de la (n-k+1)° équation pivot.

(8001) = quelconque.

(0001) = 00 (k-1) 0000

(13)	0183	20 0064 0272	(0064) = 61 $\left[M+(k-1)(n+p)\right]$ 0014
	0272	15 0001 0310	
	0310	20 0166 0322	(0166) = 61 $\left[\begin{matrix}M+k-1\\+(k-1)(n+p)\end{matrix}\right]$ 0014

en position facteur, nous avons l'adresse du dernier terme du premier membre de la (n-k+1)° équation pivot.

0322	11 0108 0216	(0108) = 00 (p) 0000
0216	15 0324 8002	(0324) = 07 9999 0252

(8002) = 69 $\left[\begin{matrix}M+k-2+p\\+(k-1)(n+p)\end{matrix}\right]$ 0266
en position facteur, nous avons l'adresse du p° second membre de la (n-k+1)° équation pivot.

8002	69 xxxx 0266	
0266	24 0017 0275	(0017)

Mémoire de travail pour les résultats intermédiaires et définitifs.

0275	69 0078 0231	(0078) = 60 $\left[M+n(n+p)\right]$ 0178
0231	24 0084 0087	(0084)

le contenu de cette mémoire sera utilisé au cours de la deuxième partie du calcul de la k° racine.

	0087	65 0222 0227 (0222) = 00 (M+n+p-1) 0000
	0227	24 0095 0148 (0095)
	0148	15 0200 0316 (0200) = 69 0000 0153
	0316	20 0014 0325 (0014) = 69 $\left[M+n+p-1\right]$ 0153

instruction pour l'appel des racines déjà calculées.

	0325	60 0064 0126	(0064)
(14)	0126	11 0166 0176	(0166)
	0176	44 0064 0080	

Tous les termes du premier membre de la (n-k+1)° équation pivot n'ont pas été utilisés. La première partie du calcul de la k° racine n'est pas terminée.

0064 61 xxxx 0014

Séquence d'Entrée dans l'A.V.F. pour effectuer une multiplication.

0014	69 xxxx 0153	
0153	24 1801 0154	
0154	69 1996 0099	VMU
0099	24 1808 0111	
0111	69 0114 1800	(0114) = instruction de Sortie

1800
Sous-Programme A. V. F.
1803

(1803) = (0114)

Séquence d'Entrée dans l'A.V.F. pour éxécuter une addition.

1803	69 0017 0020	
0020	24 1801 0204	
0204	69 1902 0005	VAA
0005	24 1808 0161	
0161	69 0164 1800	(0164) = instruction de Sortie

1800
Sous-Programme
A. V. F.
1803

(1803) = (0164)

0164 21 0017 0226 (0017)

Préparation du calcul avec le terme suivant du premier membre de la (n-k+1)° équation pivot et la racine correspondante.

0226 60 0014 0276
0276 10 0074 0229 (0074) = 00 (n+p) 0000
0229 21 0014 0326
0326 60 0064 0277
0277 10 0007 0327 (0007) = 00 0001 0000
0327 21 0064 0126 (14)

Tous les termes du premier membre de la (n-k+1)° équation pivot ont été utilisés. La première partie du calcul de la k° racine est terminé.

0080 Nous sommes prêts à aborder la "deuxième partie" du calcul de la k° racine.

REMARQUES

Nous disposons maintenant de la valeur exacte de la k° racine. C'est le contenu de la mémoire 0017.

La détermination des premières racines de chaque série ne nécessite aucun calcul. Elles sont égales aux seconds membres correspondants de la n° équation pivot. La portion de programme précédente reste cependant valable car nous avons placé le test de fin des calculs avant ceux-ci. Pour le calcul des premières racines, la première réponse au test: "44 0064 0080" sera négative.

2° PARTIE : Détermination de l'adresse de la mémoire où devra être placée la k° racine.

Pour celà nous allons chercher dans la mémoire d'adresse $[M+n(n+p)+k-1]$ associée à la (n-k+1)° équation pivot: le rang initial "l_i" de la k° inconnue. Nous comparons ensuite ce rang à ceux des inconnues déjà calculées, c'est à dire aux contenus des mémoires d'adresses consécutives:

M+n(n+p)
M+n(n+p)+1
M+n(n+p)+2
.....
M+n(n+p)+k-2

Au début de la comparaison, nous supposons que la dernière racine calculée est à placer dans la première des mémoires réservées à la série de racines en cours de calcul. Son adresse est égale à $[M+n+p-1]$. Chaque fois que nous trouvons une racine déjà calculée de rang inférieur à "l_i", nous augmentons l'adresse précédente de (n+p) et ainsi de suite ..... En fin de comparaison nous disposons de l'adresse désirée en position facteur de la mémoire 0095 dont le contenu initial est: 00 (M+n+p-1) 0000.

0080 65 0233 0137 (0233) = 69 $[M+n(n+p)]$ 0128
0137 15 0001 8002 (0001) = 00 (k-1) 0000
(8002) = 69 $\left[\begin{matrix}M+k-1\\+n(n+p)\end{matrix}\right]$ 0128
8002 69 xxxx 0128
0128 24 0206 0084 (0206) = 00 (l_i-1) 0000
(15) 0084 60 xxxx 0178 xxxx est égal initialement à M+n(n+p), puis varie en augmentant d'une unité à chaque fois.
0178 11 0206 0228 (0206)
0228 44 0281 0082

La comparaison n'est pas terminée.
Nous n'avons pas encore appelé le contenu de la mémoire $[M+n(n+p)+k-1]$ que nous savons être égal à: 00 (l_i-1) 0000.

```
0281    46 0134 0085
```

Nous rencontrons une racine déjà calculée ayant un rang initial inférieur à "l_i" (celui de la k° racine). L'adresse de la mémoire où sera stockée la k° racine, doit être augmentée de (n+p).

```
0134    60 0095 0249
0249    10 0074 0279        (0074) = 00 (n+p) 0000
0279    21 0095 0085
```

Nous venons de modifier l'adresse de la mémoire où sera stockée la k° racine;
ou bien, nous ne rencontrons pas une racine déjà calculée ayant un rang initial inférieur à "l_i".
Dans les deux cas, nous poursuivons la comparaison en cours.

```
0085    60 0084 0139        (0084) = 60 xxxx 0178
0139    10 0007 0278        (0007) = 00 0001 0000
0278    21 0084 8001  (15)
```

La comparaison est terminée.
Nous venons d'appeler le contenu de la mémoire $[M+n(n+p)+k-1]$.

```
0082    Nous sommes prêts à aborder la "troisième partie"
        du calcul de la k° racine.
```

3° PARTIE : Mise en mémoire de la k° racine.

Toutes les racines d'ordres initiaux supérieurs à "l_i" sont à déplacer. Nous effectuons ce déplacement et mettons la k° racine à sa place.

```
0082    65 0014 0328        (0014) = 69 xxxx 0153
                            avec xxxx = M+k(n+p)-1
                            il s'agit de l'adresse de la
                            mémoire où doit être stockée
                            la racine ayant le rang initial
                            le plus élevé parmi les k
                            racines déjà calculées.
0328    16 0200 0329        (0200) = 69 0000 0153
0329    10 8002 0187
0187    15 0040 0145        (0040) = 69 0000 8003
0145    16 0074 0130        (0074) = 00 (n+p) 0000
(16) 0130    11 0095 0299   (0095) = 00 xxxx 0000
                            en position facteur, nous avons
                            l'adresse de la mémoire où doit
                            être placée la k° racine.
0299    44 0180 0230
```

Le contenu de 8003 n'est pas égal à celui de la mémoire 0095.
Il faut déplacer une racine.
Nous allons placer le contenu de la mémoire dont l'adresse se trouve en position facteur de 8002, dans celle dont l'adresse se trouve en position facteur de 8003. Cette dernière mémoire est libre , et son adresse est égale à celle de la première augmentée de (n+p).

```
0180    10 8001 0189
0189    10 0192 8002        (0192) = 24 0000 0280
8002    69 xxxx 8003
8003    24 yyyy 0280        yyyy = xxxx + (n+p)
0280    11 0192 0197
0197    11 0074 0330        (0074) = 00 (n+p) 0000
0330    16 8001 0130  (16)
```

Le contenu de 8003 est égal à celui de la mémoire 0095.
Le déplacement des racines de rangs initiaux supérieurs à "l_i" est terminé. Il ne nous reste plus qu'à mettre la k° racine (calculée) à sa place.

```
0230    65 8001 0239        (8001) = (0095)
0239    15 0242 0247        (0242) = 24 0000 0331
0247    69 0017 8002        (0017) = valeur de la k° racine
8002    24 xxxx 0331
```

En opérant de cette façon, lorsque les "n" racines d'une même série auront été calculées, elles seront convenablement rangées et occuperont la place initialement occupée par la série de seconds membres correspondante.
La détermination de la k° racine étant terminée, nous abordons une série de tests.

```
0331    60 0001 0132        (0001) = 00 (k-1) 0000
0132    10 0007 0182        (0007) = 00 0001 0000
0182    21 0001 0232        k = k + 1
0232    11 0079 0283        (0079) = 00 ( n ) 0000
0283    44 0237 0038
```

Les "n" racines appartenant à la série en cours de calcul ne sont pas toutes déterminées. Nous passons au calcul de la racine suivante.

```
0237    65 0064 0282        (0064) = 61 xxxx 0014
                            à ce moment, nous avons en
                            position facteur: l'adresse
                            augmentée de l'unité du dernier
                            terme du premier membre de la
                            (n-k+2)°.équation pivot.
0282    15 0074 0332        (0074) = 00 (n+p) 0000
0332    16 0001 0333        (0001) = 00 (k-1) 0000
0333    15 0007 0183  (13)  (0007) = 00 0001 0000
                            (8002) = 61 xxxx 0014
                            en position facteur, nous avons
                            l'adresse du premier terme du
                            premier membre de la (n-k+1)°
                            équation pivot.
```

Les "n" racines appartenant à la série en cours de calcul sont toutes déterminées.

```
0038    60 0108 0184        (0108) = 00 ( p ) 0000
0184    20 0001 0234        (0001) = 00 0000 0000
0234    11 0007 0284        (0007) = 00 0001 0000
0284    44 0287 0088
```

Les séries de "n" racines correspondant aux "p" séries de seconds membres données ne sont pas toutes calculées
Nous passons au calcul de la série suivante.

```
0287    21 0108 0215  (12)
```

REMARQUE

La quantité en position facteur du contenu de la mémoire 0108 prend toutes les valeurs successives décroissantes depuis (p) jusqu'à 0001. A chaque valeur correspond le calcul d'une série de racines.

Les "p" séries de "n" racines ont toutes été calculées.
La résolution du système linéaires est terminée.
Les résultats convenablement rangés occupent la place initialement réservées aux "p" séries de seconds membres données.

```
0088    Nous effectuons l'instruction placée en 8001
        lors de l'ENTREE dans le SOUS PROGRAMME de
        RESOLUTION de SYSTEMES LINEAIRES.
```

1, 4 SOUS-PROGRAMME de RESOLUTION de SYSTEMES LINEAIRES en SIMPLE PRECISION. METHODE de JORDAN Itérations. (Référence J I S P).

1, 4-1 NOTICE d'UTILISATION

Objet : Etant donné, en simple précision, la matrice et les vecteurs seconds membres d'un système linéaire, ce sous-programme calcule en virgule flottante *Flair* et en simple précision, les vecteurs solutions.

Conditions d'utilisation : Si « n » désigne l'ordre de la matrice du système linéaire, et « p » le nombre de vecteurs seconds membres, la limite d'utilisation de ce sous-programme est :

$$2\ n\ (n + p) + p\ n \leq 1400$$

Ainsi avec $p = 1$, nous avons $n_{MAX} = 25$.

La virgule flottante simulée étant du type *Flair*, on représentera

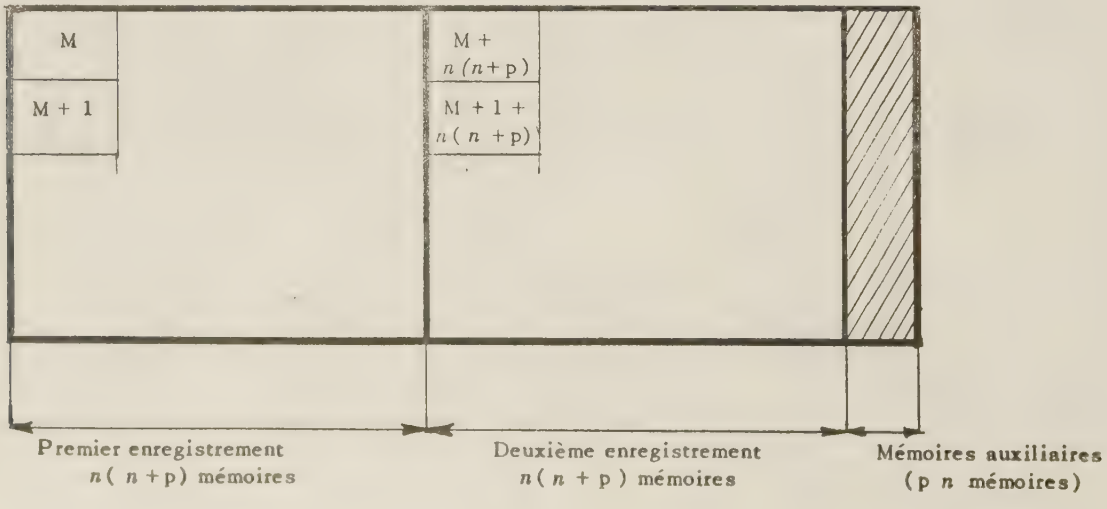
1 par 51 1000 0000 +

Il faudra enregistrer l'A.V.F. (Arithmétique en virgule flottante Simple Précision) sur le tambour.

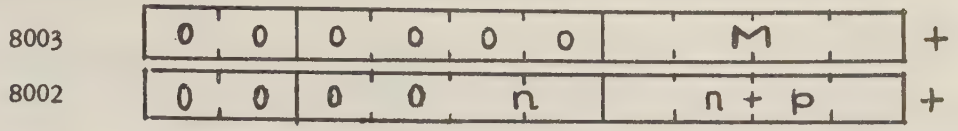
Utilisation : Le système à résoudre doit être rangé *deux fois*, ligne par ligne, dans des mémoires consécutives du tambour. Chaque ligne se composant des « n » coefficients du premier membre et des « p » coefficients des seconds membres, c'est-à-dire d'une équation complète du système. Le premier enregistrement des données utilise un bloc de $[n\ (n + p)]$ mémoires consécutives à partir d'une adresse M fixée par l'utilisateur. Le deuxième enregistrement utilise un autre bloc de $[n\ (n + p)]$ mémoires consécutives à partir de l'adresse $[M + n\ (n + p)]$.

Le sous-programme utilise encore « $p\ n$ » mémoires auxiliaires pour les résultats intermédiaires. Ces mémoires auxiliaires sont situées immédiatement à la suite du deuxième enregistrement des données. Elles immobilisent donc un bloc de $p\ n$ mémoires consécutives à partir de l'adresse $[M + 2\ n\ (n + p)]$.

Nous aurons ainsi le schéma suivant :



Avec les notations déjà définies, pour entrer dans le sous-programme de résolution de systèmes linéaires, l'utilisateur devra placer en :



une instruction de sortie.

Après les calculs, les "p n" mémoires auxiliaires contiennent les «p» vecteurs solutions correspondant aux vecteurs seconds membres donnés. Ainsi, le premier vecteur solution (correspondant au premier vecteur second membre) occupe n mémoires consécutives à partir de l'adresse $[M + 2n(n + p)]$. De même, le i^{e} vecteur solution occupe n mémoires consécutives à partir de l'adresse $[M+2n(n + p)+(i-1)n]$.

Nous pouvons très bien avoir $p > n$, la seule limite étant celle dûe à la formule du paragraphe précédent.

Exemple : supposons $n = 6$, $p = 2$ et $M = 0125$. Les contenus de 8003 et 8002 à l'entrée seront les suivants :

8003 00 0000 0125 +

8002 00 0006 0008 +

Le premier enregistrement des données (matrice et vecteurs seconds membres) occupe les mémoires 0125 à 0172. Le deuxième enregistrement des données immobilise les mémoires 0173 à 220. Ainsi la 3° ligne du système sera enregistrée : une première fois dans les mémoires 0141 à 0148 et une seconde fois dans les mémoires 0189 à 0196. Les mémoires 0221 à 0232 seront les mémoires auxiliaires pour les résultats intermédiaires. Les résultats occuperont pour :

- la première série de seconds membres (premier vecteur solution), les mémoires :0221, 0222, 0223, 0224, 0225 0226.
- la seconde série de seconds membres, les mémoires : 0227, 0228, 0229, 0230, 0231, 0232.

Forme du sous-programme : l'utilisateur dispose de deux paquets decartes dont les contenus devront être enregistrés sur le tambour :

1°/ *l'A V F* (Arithmétique Virgule Flottante simple Précision). Ce programme occupe les mémoires 1800 à 1999 sauf les zones entrée et sortie standard de la bande 1950 (1950 à 1960 et 1977 à 1986).

Il se présente sous forme decartes «chargement 7 mots» (cartes roses) et est précédé d'un programme de chargement (cartes jaunes) qui utilise la bande 0000. Il faudra donc prévoir son chargement en temps utile dans le cas où l'on utiliserait la bande 0000 pour stocker les données ou à toute autre fin.

2°/ *Le SOUS-PROGRAMME* de Résolution de Systèmes Linéaires. Il se présente sous forme numérique translatable «1 mot par carte» (cartes blanches). Le facteur de translation Δ qui peut être quelconque mais inférieur ou égal à 1400, sera chargé en 1950.

La première instruction de cesous-programme se trouve en $\Delta + 0000$.

Encombrement : 200 mémoires pour l'A.V.F. (obligatoirement de 1800 à 1999, les zones entrée et sortie standard de la bande 1950 étant disponibles).

400 mémoires pour le sous-programme.

Arrêt programmé : Lorsque la matrice du système est singulière, nous avons l'arrêt :

01 0000 9999

Méthode utilisée : La méthode utilisée est celle de JORDAN Itérations qui procède de la façon suivante :

- Résolution du système linéaire proposé ($A\,x = b$) à l'aide de la méthode de JORDAN classique, le pivot étant pris sur la diagonale principale de la matrice du système. Soit x_0 (vecteur ou matrice) la solution ainsi calculée.

- Amélioration de la solution x_0 en lui ajoutant une correction ε_0 (vecteur ou matrice) obtenue par la résolution du système proposé dans lequel les résidus relatifs à la solution x_0 ($b - Ax_0$) ont été substitués aux seconds membres donnés. Soit $x_1 = x_0 + \varepsilon_0$ la nouvelle solution obtenue qui sera améliorée de la même façon que la solution x_0 et ainsi de suite.....

La détermination de chaque nouvelle solution constitue « une itération ». Lorsque les résultats, en virgule flottante simple précision, de deux itérations successives sont identiques, les calculs sont terminés.

Remarques : A la fin de la résolution d'un système linéaire :

- le nombre d'itérations effectuées se trouve en position « instruction suivante » du contenu de la mémoire $\Delta + 0185$.

- le coefficient maximum en valeur absolue, de la correction effectuée au cours de la dernière itération se trouve avec son signe et en virgule flottante *Flair*, dans la mémoire $\Delta + 0211$.

D'autre part, au cours de la résolution d'un système linéaire, on pourra connaître le nombre d'itérations déjà effectuées en prévoyant un *Arrêt Prédéterminé* en $\Delta + 0187$. Le nombre d'itérations se trouve alors en position « instruction suivante » du contenu de 8003.

I, 4-2 NOTE EXPLICATIVE

(voir page suivante)

Nous appellerons "n" l'ordre du système linéaire à résoudre, et "p" le nombre de vecteurs seconds membres.

Le système linéaire est rangé DEUX FOIS, équation par équation, dans des mémoires consécutives du tambour. Chaque équation comprend les "n" coefficients du premier membre et les "p" seconds membres. Le premier enregis - trement des données utilise un bloc de $[n(n + p)]$ mémoires consécutives à partir d'une adresse "M". Le deuxième enregis - trement utilise un second bloc de $[n(n + p)]$ mémoires consé - cutives à partir de l'adresse $[M + n(n + p)]$. Enfin, "pn" mémoires consécutives à partir de l'adresse $[M + 2n(n + p)]$ sont réservées pour le stockage des approximations successives des "p" vecteurs solutions associés aux vecteurs seconds membres donnés.

Pour entrer dans le Sous-Programme, il faut charger 8001, 8002, et 8003 de la façon suivante:

(8001) = instruction de sortie
(8002) = 00 (n) (n + p)
(8003) = 00 0000 (M)

La première instruction du Sous-Programme se trouve en 0000.

REMARQUE

Pour faciliter l'étude de ce Sous-Programme, nous l'avons lui-même décomposé en une série de Sous-Programmes dont l'enchaînement est assuré par un Programme de Liaison.

Nous étudions en premier lieu ce Programme de Liaison, les différents Sous-Programmes étant étudiés ensuite .

MEMOIRES de TRAVAIL

1°) à contenus constants

(0036) = 00 (n) 0000
(0042) = 00 (n + p) 0000
(0086) = 00 (pn) 0000
Nombre des composantes des "p" vecteurs solutions recherchés.
(0093) = 00 (p) 0000
(0103) = 00 (M) 0000
Première adresse du bloc de $(2n^2 + 3pn)$ mémoires consécutives, nécessaire pour le double enregis - trement des données et le stockage des résultats intermédiaires et définitifs.
(0142) = 00 $[n(n + p)]$ 0000
Nombre de mémoires occupées par un seul enregistrement des données (matrice du système linéaire d'ordre n, et "p" vecteurs seconds membres) .
(0157) = 00 $[M + 2n(n + p)]$ 0000
Adresse de la première des "pn" mémoires consécutives prévues pour le stockage des résultats intermédiaires et définitifs (approximations successives des "p" vecteurs solutions).

2°) à contenus variables.

(0001) = 60 yyyy 0100
Ce contenu est utilisé dans le Sous-Programme de DIVISION de l'EQUATION PIVOT par le PIVOT pour tester la fin des divisions.
Au cours de chacune des "n" éliminations, le

symbole "yyyy" représente l'adresse, augmentée de l'unité, du dernier second membre de l'équation pivot.

= Mot d'ORIENTATION utilisé pendant la permutation du double enregistrement de deux équations.
Nous aurons:

(0001) = 60 0142 0337

pendant la permutation des premiers enregistrements,
et (0001) = 65 0070 0026
pendant la permutation des deuxièmes enregistre - ments.

= 61 wwww 0050
Ce contenu est utilisé dans le Sous-Programme d'ELIMINATION pour tester la fin du traitement des (n - 1) équations autres que l'équation pivot.
Au cours de chacune des "n" éliminations, le symbole "wwww" représente l'adresse, augmentée de l'unité, du dernier second membre de l'équation pivot.

= 60 yyyy 0014 ou 60 yyyy 0305
Ces contenus sont utilisés dans le Sous-Programme de CALCUL des APPROXIMATIONS pour tester la fin du transfert, ou de la correction, des composantes de rang "k_c" des "p" vecteurs solutions.
Nous aurons:

(0001) = 60 yyyy 0014

pendant la première résolution du système linéaire,
et (0001) = 60 yyyy 0305
pendant les résolutions suivantes ou ITERATIONS.
Dans les deux cas, le symbole "yyyy" représente l'adresse de la composante de rang "k_c" du dernier vecteur solution, ou du dernier vecteur correction.

= 61 zzzz 0007
Ce contenu est utilisé dans le Sous-Programme de CALCUL des RESIDUS pour tester la fin du calcul de la composante de rang "k" du i° vecteur résidu.

Le symbole "zzzz" représente l'adresse, augmentée de l'unité, du dernier coefficient du premier membre de la k° équation.

(0007) = 21 xxxx 0259
Instruction utilisée dans le Sous-Programme de DIVISION de l'EQUATION PIVOT par le PIVOT pour la mise en mémoire des quotients.
Initialement, le symbole "xxxx" représente l'adresse:
- du terme de rang (k + 1) dans le premier membre de la k° équation (équation pivot), dans le cas de la Première Résolution du système linéaire.
- du premier second membre de l'équation pivot, dans le cas des autres résolutions (Itérations).

= 00 (1 - k) 0000
pendant la permutation des premiers enregistrements de deux équations,
= 00 (n^2 + pn + 1 - k) 0000
pendant la permutation des deuxièmes enregistre - ments des mêmes équations que ci-dessus.
Ces deux permutations ont lieu, quand elles s'avèrent nécessaires, au début de la k° élimination.

= 69 qqqq 0003
Instruction utilisée dans le Sous-Programme d'ELIMINATION pour appeler les coefficients des (n - 1) équations autres que l'équation pivot.
Initialement: qqqq = M + k.

= 69 zzzz 0003
Instruction utilisée dans le Sous-Programme de CALCUL des APPROXIMATIONS pour appeler les composantes, de rang k_c, des "p" vecteurs solutions appartenant à la dernière approximation calculée.
Initialement: zzzz = M + 2n(n + p) + k_c - 1.

= 69 wwww 0053
Instruction utilisée dans le Sous-Programme de CALCUL des RESIDUS pour appeler les "n" composantes du i^e vecteur solution.
Initialement: wwww = M + 2n(n + p) + n(i - 1).

(0010) = 00 (k) 0000
Au cours de chaque résolution de système linéaire, "k" représente le rang de l'inconnue que nous nous préparons à éliminer.
Nous aurons: $1 \leqslant k \leqslant n$.

(0014) = Valeur exacte du pivot pendant toute la durée du Sous-Programme de DIVISION de l'EQUATION PIVOT par le PIVOT.

= Multiplicateur associé à l'équation en cours de traitement à l'aide du Sous-Programme d'ELIMINATION.

= 21 zzzz 0353
Instruction utilisée dans le Sous-Programme de CALCUL des APPROXIMATIONS pour la mise en mémoire des composantes, de rang k_c, des "p" vecteurs solutions.
Initialement: zzzz = M + 2n(n + p) + k_c - 1 .

(0028) = 00 (d) 0000
Au cours du Sous-Programme de DIVISION de l'EQUATION PIVOT par le PIVOT, lorsque le terme de rang k du premier membre de la k^e équation (le pivot) est nul, "d" représente le nombre d'équations déjà examinées parmi les (n - k) dernières équations pouvant encore fournir un pivot.
Nous aurons donc: $1 \leqslant d \leqslant (n - k)$.

= 00 (k_t) 0000
Au cours du Sous-Programme d'ELIMINATION, "k_t" représente le numéro d'ordre de l'équation en cours de traitement.
Nous aurons: $1 \leqslant k_t \leqslant n$.

= Compteur des composantes des vecteurs solutions, non affectées par les corrections effectuées à l'aide du Sous-Programme de CALCUL des APPROXIMATIONS.

= 00 (i - 1) 0000
avec "i" représentant le numéro d'ordre du vecteur résidu en cours de détermination à l'aide du Sous-Programme de CALCUL des RESIDUS.
Nous aurons: $1 \leqslant i \leqslant p$

(0035) = 00 xxxx 0000
Dans le Sous-Programme d'ELIMINATION, "xxxx" représente l'adresse du terme de rang (k + 1) dans le premier membre de la k_t^e équation.
Initialement: xxxx = M + k.

= Valeur exacte de la dernière correction utilisée dans le Sous-Programme de CALCUL des APPROXIMATIONS.

(0037) = 00 0000 (n + p)
au début du Programme de LIAISON.

= 60 xxxx 0100
Instruction utilisée dans le Sous-Programme de DIVISION de l'EQUATION PIVOT par le PIVOT pour appeler tous les coefficients de l'équation pivot (dividendes).
Initialement: xxxx = M + (k - 1)(n + p + 1) + 1.

= 60 yyyy 0322
Au cours du Sous-Programme de DIVISION de l'EQUATION PIVOT par le PIVOT, lorsque le pivot est nul, cette instruction permet d'appeler tous les coefficients susceptibles d'être pris comme "pivot".

= 61 vvvv 0050
Instruction utilisée dans le Sous-Programme d'ELIMINATION pour appeler les coefficients de l'équation pivot.
Initialement: vvvv = M + (k - 1)(n + p + 1) + 1.

= 60 xxxx 0014 ou 60 xxxx 0305
Ces instructions sont utilisées dans le Sous-Programme de CALCUL des APPROXIMATIONS pour appeler les composantes, de rang k_c, des vecteurs solutions, ou des vecteurs corrections.
Nous aurons:
(0037) = 60 xxxx 0014
pendant la première résolution du système linéaire,
et (0037) = 60 xxxx 0305
pendant les résolutions suivantes ou ITERATIONS.
Dans les deux cas, nous avons initialement:
xxxx = M + n .

= 61 yyyy 0007
Instruction utilisée dans le Sous-Programme de CALCUL des RESIDUS pour appeler les coefficients du premier membre de la k° équation (deuxième enregistrement) afin de calculer la k° composante d'un vecteur résidu.
Initialement: k = 1 et yyyy = M + n(n + p).

(0043) = 00 (n + p - k) 0000
au cours de chaque résolution de système linéaire.

= Instruction utilisée au cours du Sous-Programme d'ADDITION en VIRGULE FLOTTANTE DOUBLE PRECISION pour aligner le cadrage du plus petit nombre sur celui du plus grand.

(0044) = Mot d'ORIENTATION utilisé par le Sous-Programme de CALCUL des APPROXIMATIONS.

(0051) = Instruction de SORTIE du Sous-PROGRAMME de RESOLUTION de SYSTEMES LINEAIRES.

(0062) = Dix derniers chiffres de la mantisse d'un nombre exprimé en Virgule Flottante Double Précision.
Cette mémoire est utilisée dans les Sous-Programmes de CALCUL des RESIDUS et d'ADDITION en VIRGULE FLOTTANTE DOUBLE PRECISION.

(0064) = 21 qqqq 0203
Instruction utilisée dans le Sous-Programme d'ELIMINATION pour la mise en mémoire des nouveaux coefficients des (n - 1) équations autres que l'équation pivot.
Initialement: qqqq = M + k.

= 11 8001 0207
pendant le déroulement du Sous-Programme de CALCUL des APPROXIMATIONS.

= 21 xxxx 0354
Instruction utilisée dans le Sous-Programme de CALCUL des RESIDUS pour la mise en mémoire des "n" composantes de chacun des "p" vecteurs résidus.
Initialement: xxxx = M + n .

(0070) = 00 [M + (k - 1)(n + p + 1)] 0000
Pendant toute la durée de la résolution d'un système linéaire, cette mémoire contient l'adresse du pivot de l'élimination en cours. Ici, nous avons l'adresse du pivot de la k° élimination.

= Pendant le déroulement du Sous-Programme d'ADDITION en VIRGULE FLOTTANTE DOUBLE PRECISION, cette mémoire contient successivement:
- la différence des exposants des deux nombres à additionner, si l'exposant du nombre B est supérieur à celui du nombre A .
- une instruction utilisée pour aligner le cadrage du plus petit nombre sur celui du plus grand.
- le nombre de zéros détectés à gauche du premier chiffre significatif de la mantisse du résultat.

(0074) = En position "instruction suivante", l'exposant d'un nombre exprimé en Virgule Flottante Double Précision.

Cette mémoire est utilisée dans les Sous-Programmes de CALCUL des RESIDUS et d'ADDITION en VIRGULE FLOTTANTE DOUBLE PRECISION.

(0085) = 00 (k_c - 1) 0000
Avec "k_c" représentant le rang des composantes des vecteurs solutions que nous nous préparons à corriger à l'aide du Sous-Programme de CALCUL des APPROXIMATIONS.
Nous aurons: $1 \leqslant k_c \leqslant n$

= 00 (k - 1) 0000
Avec "k" représentant le rang de la composante, en cours de détermination à l'aide du Sous-Programme de CALCUL des RESIDUS, appartenant au i° vecteur résidu.
Nous aurons: $1 \leqslant k \leqslant n$ $(1 \leqslant i \leqslant p)$.

(0152) = Neuf premiers chiffres, précédés d'un zéro, de la mantisse d'un nombre exprimé en Virgule Flottante Double Précision.
Cette mémoire est utilisée dans les Sous-Programmes de CALCUL des RESIDUS et d'ADDITION en VIRGULE FLOTTANTE DOUBLE PRECISION.

(0154) = Mot d'ORIENTATION utilisé par le Sous-Programme d'ELIMINATION.

(0185) = Comptage des ITERATIONS.
Ce comptage se fait en position "instruction suivante".

(0211) = Valeur exacte de la correction maximum en valeur absolue, rencontrée dans le Sous-Programme de CALCUL des APPROXIMATIONS.

= 00 0000 0000 ou 00 0000 0010
Le contenu de cette mémoire participe au calcul de l'exposant du résultat fourni par le Sous-Programme d'ADDITION en VIRGULE FLOTTANTE DOUBLE PRECISION.

(0348) = Mot d'ORIENTATION utilisé par le Sous-Programme de DIVISION de l'EQUATION PIVOT par le PIVOT.

(0365) = Instruction de SORTIE du Sous-Programme de CALCUL des RESIDUS.
Le chargement de cette mémoire est assuré par le Programme de LIAISON.

(1803) = Instruction de SORTIE du Sous-Programme A.V.F. (Arithmétique en Virgule Flottante Simple Précision du type FLAIR).

(1831) = Dix derniers chiffres de la mantisse d'un nombre exprimé en Virgule Flottante Double Précision.
Voir la REMARQUE ci-dessous.

(1879) = En position "instruction suivante", l'exposant d'un nombre exprimé en Virgule Flottante Double Précision.
Voir la REMARQUE ci-dessous.

(1890) = Dix premiers chiffres de la mantisse d'un nombre exprimé en Virgule Flottante Double Précision.
Voir la REMARQUE ci-dessous.

REMARQUE

Ces trois dernières mémoires de travail appartiennent au Sous-Programme A.V.F.

Au cours du Sous-Programme de CALCUL des RESIDUS, elles contiennent le produit, exprimé en Virgule Flottante Double Précision, de deux nombres exprimés en Virgule Flottante Simple Précision.
Ce produit est effectué à l'aide du Sous-Programme A.V.F. convenablement modifié.

PROGRAMME de LIAISON

Ce programme débute par le calcul d'une série de constantes, fonctions de M, n et p, qui sont utilisées dans les différents Sous-Programmes.

0000	24 0051 0359	(0051) = instruction de sortie
0359	69 0018 0121	(0018) = 00 0001 0000
0121	22 0036 0389	(0036) = 00 (n) 0000
0389	69 8003 0302	(8003) = 00 0000 (M)
0302	23 0037 0244	(0037) = 00 0000 (n + p)
0244	35 0004 0212	
0212	21 0103 0262	(0103) = 00 (M) 0000
0262	22 0042 0352	(0042) = 00 (n) 0000
0352	60 0037 0294	
0294	19 0036 0312	
0312	20 0142 0362	(0142) = 00 (n^2 + pn) 0000
0362	15 8001 0171	
0171	15 0103 0213	
0213	20 0157 0263	(0157) = 00 (M+2n(n+p)) 0000
0263	60 0042 0313	
0313	11 0036 0344	
0344	21 0093 0363	(0093) = 00 (p) 0000
0363	19 0036 0364	
0364	30 0004 0176	
0176	20 0086 0394	(0086) = 00 (pn) 0000

Première Résolution du Système Linéaire.

Nous préparons maintenant les entrées dans les différents Sous-Programmes qui vont nous permettre d'effectuer la première résolution du système linéaire.

Nous travaillerons sur le premier enregistrement du système, qui occupe $[n(n + p)]$ mémoires consécutives à partir de l'adresse M . La méthode utilisée sera celle de JORDAN, le pivot étant pris sur la diagonale principale de la matrice du système.

Par ailleurs, au cours de cette première élimination, il nous faut conserver en mémoires:

- les valeurs exactes des "n" pivots
- les "n" séries de (n - 1) multiplicateurs utilisés pour l'élimination des "n" inconnues.

	0394	69 0316 0269	(0316) = 69 0369 0022
	0269	24 0365 0118	(0365) = Instruction de Sortie du Sous-Programme de Calcul des Résidus.
	0118	60 0221 0226	(0221) = 60 0000 0014
	0226	15 0300 0366	(0300) = 65 0035 0039
	0366	69 0252 0168	(0252) = 20 0001 0037
	0168	24 0348 0218	(0348) = Mot d'Orientation du Sous-Programme de Division de l'Equation Pivot par le Pivot.
	0218	21 0044 0268	(0044) = Mémoire de Travail du Sous-Programme de Calcul des Approximations.
	0268	20 0154 0318	(0154) = Mot d'Orientation du Sous-Programme d'Elimination.
	0318	66 0018 0276	(0018) = 00 0001 0000
	0276	24 0010 0368	(0010) = 00 (k) 0000 avec "k" représentant le rang de l'inconnue que l'on va éliminer. Initialement: k = 1.
	0368	15 0042 0319	(0042) = 00 (n + p) 0000
	0319	10 0103 0165	(0103) = 00 (M) 0000
(20)	0165	21 0070 0273	(0070) = 00 $\left[\frac{M+}{(k-1)(n+p+1)}\right]$ 0000 en position facteur, nous avons l'adresse du pivot.
	0273	20 0043 0146	(0043) = 00 (n + p - k) 0000

Nous sommes prêts à commencer la k° élimination.

0146

Sous-Programme
de
DIVISION
de
l'EQUATION PIVOT
par le PIVOT

0250

0250

Sous-Programme
d'ELIMINATION

0046

La k° élimination est terminée.

```
0046    60 0010 0115        (0010) = 00 ( k ) 0000
0115    11 0036 0141        (0036) = 00 ( n ) 0000
0141    44 0095 0096
```

La première résolution du système linéaire n'est pas terminée. Nous préparons l'élimination suivante.

```
0095    10 8001 0303
0303    10 0018 0173        (0018) = 00 0001 0000
0173    21 0010 0013        (0010) = 00 ( k ) 0000
                            Nous venons de faire:
                                   k = k + 1
0013    60 0070 0075        (0070)
                            en position facteur, nous
                            avons l'adresse du pivot de
                            la dernière élimination.
0075    10 0018 0223
0223    10 0042 0147        (0042) = 00 (n + p) 0000
0147    15 8001 0205
0205    16 0010 0165  (20)  (8002) = 00 (n + p - k) 0000
                            (8003)
                            en position facteur, nous
```

avons l'adresse du pivot de
la prochaine élimination.

Les "n" éliminations ont été effectuées. La PREMIERE RESOLUTION du système linéaire est terminée.

Etant donnés les modes opératoires des Sous-Programmes de Division de l'Equation Pivot par le Pivot et d'Elimination, nous avons maintenant, dans les $[n(n + p)]$ mémoires consécutives à partir de l'adresse "M":

1°) à la place de la matrice du système, qui a été détruite:

a) les "n" pivots occupant les mémoires qui contenaient initialement les coefficients de la diagonale principale de la matrice.

b) les "n" séries de (n - 1) multiplicateurs associées aux "n" éliminations qui occupent les autres mémoires.

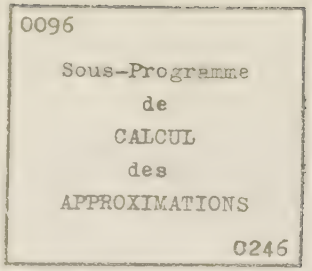
2°) les vecteurs solutions du système linéaire, à la place initialement occupée par les "p" vecteurs seconds membres correspondants.

REMARQUES

Les (n - 1) multiplicateurs associés à l'élimination de la k° inconnue occupent les (n - 1) mémoires qui contenaient initialement les coefficients de la k° inconnue des équations du système, autres que l'équation pivot (la k° équation). Le pivot de la k° élimination se trouve dans la mémoire d'adresse $[M + (k - 1)(n + p + 1)]$, initialement occupée par le coefficient de la k° inconnue de la k° équation.

D'autre part, le deuxième enregistrement des données n'a subi aucune modification.

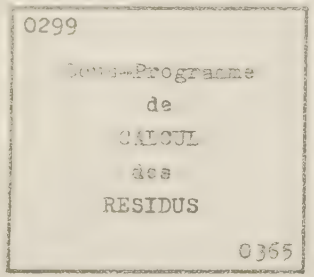
Avant de poursuivre les calculs, à l'aide du Sous-Programme de Calcul des Approximations, nous allons transférer les "p" vecteurs solutions dans les "pn" mémoires consécutives, réservées à cet effet, à partir de l'adresse $[M + 2n(n + p)]$.



0246 60 0028 0183 (0028) = 00 0000 0000
Le contenu de la mémoire 0028 est obligatoirement nul lors de la première sortie du Sous-Programme de Calcul des Approximations après la première résolution du système.

0183 11 0086 0341 (0086) = 00 (pn) 0000
0341 44 0299 0051

Nous allons calculer les vecteurs résidus relatifs à la première approximations des "p" vecteurs solutions.



La première résolution du système linéaire est terminée; nous commençons les ITERATIONS.

Préparation des entrées dans les différents Sous-Programmes qui vont nous permettre d'effectuer les calculs des corrections successives à apporter à la première approximation des "p" vecteurs solutions.

0365 69 0369 0022 (0369) = 60 0185 0072
0022 24 0365 0220 (0365) = Instruction de Sortie du Sous-Programme de Calcul des Résidus.
0220 60 0326 0281 (0326) = 60 0000 0305
0281 20 0185 0338 (0185) = 00 0000 0000
Remise à zéro du compteur d'ITERATIONS.
0338 15 0270 0376 (0270) = 16 0093 0320
0376 69 0229 0168 (0229) = 20 0001 0371
0168 24 0348 0218 (0348) = Mot d'Orientation du Sous-Programme de Division de l'Equation Pivot par le Pivot.
0218 21 0044 0268 (0044) = Mémoire de Travail du Sous-Programme de Calcul des Approximations.
0268 20 0154 0318 (0154) = Mot d'Orientation du Sous-Programme d'Elimination.

La résolution d'un nouveau système linéaire peut commencer. La matrice de ce système est la même que celle du système initial, traité au cours de la première résolution. Les seconds membres sont les "p" vecteurs résidus associés à la dernière approximation des vecteurs solutions.

En utilisant les pivots et les multiplicateurs mis en mémoires au cours de la première résolution, les calculs se limiteront aux seuls seconds membres. La méthode utilisée est obligatoirement celle de JORDAN, le pivot étant pris sur la diagonale principale de la matrice du système.

Les pivots et multiplicateurs occupent les mémoires initialement réservées à la matrice du système. (voir "Première Résolution du Système Linéaire"). Les vecteurs résidus occupent la place des seconds membres initiaux.

(22) 0318 66 0018 0276 (0018) = 00 0001 0000
0276 24 0010 0368 (0010) = 00 (k) 0000

avec "k" représentant le rang de l'inconnue que l'on va éliminer.
Initialement: k = 1.

```
0368    15 0042 0319        (0042) = 00 (n + p) 0000
0319    10 0103 0165        (0103) = 00 (  M  ) 0000
(21) 0165    21 0070 0273   (0070) = 00 [M + (k-1)(n+p+1)] 0000
```

en position facteur, nous avons l'adresse du pivot.

```
0273    20 0043 0146        (0043) = 00 (n + p - k) 0000
```

Nous sommes prêts à commencer la k° élimination.

0146
Sous-Programme de DIVISION de l'EQUATION PIVOT par le PIVOT
0250

0250
Sous-Programme d' ELIMINATION
0046

La k° élimination est terminée.

```
0046    60 0010 0115        (0010) = 00 (  k  ) 0000
0115    11 0036 0141        (0036) = 00 (  n  ) 0000
0141    44 0095 0096
```

La résolution du système linéaire n'est pas terminée.
Nous préparons l'élimination suivante.

```
0095    10 8001 0303
0303    10 0018 0173        (0018) = 00 0001 0000
0173    21 0010 0013        (0010) = 00 (  k  ) 0000
```

Nous venons de faire:

$$k = k + 1$$

```
0013    60 0070 0075        (0070)
```

en position facteur, nous avons l'adresse du pivot de la dernière élimination.

```
0075    10 0018 0223        (0018) = 00 0001 0000
0223    10 0042 0147        (0042) = 00 (n + p) 0000
0147    15 8001 0205
0205    16 0010 0165 (21)   (8002) = 00 (n + p - k) 0000
                            (8003)
```

en position facteur, nous avons l'adresse du pivot de la prochaine élimination.

Les "n" éliminations ont été effectuées.
Les solutions obtenues, qui occupent les mémoires initialement réservées aux "p" séries de seconds membres, constituent les corrections à apporter à la dernière approximation des "p" vecteurs solutions.

0096
Sous-Programme de CALCUL des APPROXIMATIONS
0246

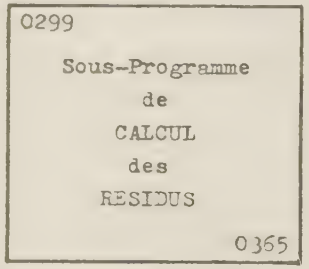
```
0246    60 0028 0183        (0028) = 00 nnnn 0000
```

avec "nnnn" représentant le nombre de composantes des "p" vecteurs solutions qui n'ont pas été affectées par les dernières corrections.

```
0183    11 0086 0341        (0086) = 00 ( pn ) 0000
```

0341 44 0299 0051

Il faut poursuivre les ITERATIONS.
Nous préparons l'itération suivante en calculant les vecteurs résidus relatifs à la dernière approximation des "p" vecteurs solutions.



0365 60 0185 0072 (0185)
en position instruction, nous avons le nombre d'itérations moins un .
0072 10 0032 0187 (0032) = 00 0000 0001
0187 21 0185 0318 (22)

Les dernières corrections calculées n'ont aucune influence sur les "p" vecteurs solutions; il est inutile de poursuivre les ITERATIONS.

0051 Nous effectuons l'instruction placée en 8001 lors de l'ENTREE dans le SOUS PROGRAMME de RESOLUTION de SYSTEMES LINEAIRES.
Nous disposons des résultats sous la forme indiquée dans la NOTICE d'UTILISATION de ce Sous-Programme.

E T U D E
d e s
S O U S P R O G R A M M E S

SOUS PROGRAMME de DIVISION de l'EQUATION PIVOT par le PIVOT.

But du Sous-Programme

Nous abordons la k° élimination au cours de laquelle sera éliminée la k° inconnue. L'équation pivot est la k° équation (les inconnues sont éliminées dans l'ordre, le pivot étant pris sur la diagonale principale de la matrice du système) . Elle occupe $(n + p - k + 1)$ mémoires consécutives à partir de l'adresse $\left[M + (k - 1)(n + p + 1)\right]$, qui est celle du pivot.

Il faut d'abord vérifier que le pivot est non nul. Dans le cas contraire, un autre pivot sera recherché, et l'équation l'ayant fourni sera permutée avec la k° équation.

Ensuite deux cas sont à considérer.

1°) Il s'agit de la première résolution du système linéaire, tous les termes de l'équation pivot, sauf le pivot, sont divisés par le pivot.

2°) A partir de la deuxième résolution du système linéaire, seuls les seconds membres de l'équation pivot sont divisés par le pivot.

Dans les deux cas, les quotients obtenus sont conservés en mémoire, à la place occupée par les dividendes dont ils sont issus. Le pivot reste inchangé pendant toutes les résolutions et occupe toujours la même mémoire d'adresse $\left[M + (k-1)(n+p+1)\right]$.

Entrée dans le Sous-Programme

Les contenus de 8001, 8002 et 8003 pouvant être quelconques, seules quatre mémoires devront être convenablement chargées.

(0010) = 00 (k) 0000
(0043) = 00 (n + p − k) 0000
(0070) = 00 $\left[M + (k-1)(n+p+1)\right]$ 0000
(0348) = Mot d'Orientation
= (0252) = 20 0001 0037
lors de la première résolution.
= (0229) = 20 0001 0371
à partir de la deuxième résolution.

D'autre part, il sera fait appel à un certain nombre de constantes chargées au tout début du Programme de Liaison.

(0036) = 00 (n) 0000
(0042) = 00 (n + p) 0000
(0093) = 00 (p) 0000
(0142) = 00 (n^2+pn) 0000

Sortie du Sous-Programme

Elle est fixe. Nous sortons du Sous-Programme de Division de l'Equation Pivot par le Pivot, pour entrer dans celui d'Elimination.

0146	65 0070 0026	(0070) = 00 $\left[\frac{M+}{(k-1)(n+p+1)}\right]$ 0000
0026	15 0129 8002	(0129) = 60 0000 0209
8002	60 xxxx 0209	(8003) = valeur exacte du pivot.
0209	44 0163 0314	

Le pivot n'est pas nul.

0163	24 0014 0068	
0068	65 0070 0076	(0070)
0076	15 0179 0284	(0179) = 60 0001 0100
0284	20 0037 0340	(0037) = 60 xxxx 0100 avec "xxxx" égal à l'adresse du premier terme à diviser.
0340	69 0343 0396	(0343) = 21 0000 0259
0396	22 0007 0310	(0007) = 21 xxxx 0259 Instruction pour le stockage des quotients.
0310	15 0043 0348	(0043) = 00 (n + p − k) 0000 (0348) = Mot d'Orientation.

Premier cas: Il s'agit de la première résolution du système linéaire.

	0348	20 0001 0037	(0001) = 60 yyyy 0100 avec "yyyy" égal à l'adresse, augmentée de l'unité, du dernier terme à diviser.
(1)	0037	60 xxxx 0100	(8003) = dividende

Séquence d'Entrée dans l'A.V.F. pour effectuer une division.

0100	69 1900 0049	VDV
0049	24 1808 0011	
0011	69 0014 0017	(0014) = valeur exacte du pivot
0017	24 1801 0004	
0004	69 0007 1800	(0007) = instruction de Sortie

1800	
Sous-Programme A. V. P.	
1803	(1803) = (0007)

Mise en mémoire du quotient

1803	21 xxxx 0259	Le quotient est mis à la place du dividende.

Préparation de la division suivante.

	0259	65 0037 0393	(0037) = 60 xxxx 0100 avec "xxxx" égal à l'adresse du dernier terme divisé.
	0393	15 0018 0126	(0018) = 00 0001 0000
(2)	0126	69 0343 0398	(0343) = 21 0000 0259
	0398	22 0007 0360	(0007) = 21 xxxx 0259 Instruction pour le stockage des quotients.
	0360	20 0037 0390	
	0390	16 0001 0309	(0001) = 60 yyyy 0100 voir page précédente
	0309	45 0037 0250	

Toutes les divisions ne sont pas terminées.

0037	Nous allons effectuer une nouvelle division. Voir en (1).

Toutes les divisions ont été effectuées.

0250	Nous sortons du Sous-Programme de Division de l'Equation Pivot par le Pivot pour entrer dans celui d'Elimination.

Deuxième cas: Il s'agit de la deuxième résolution ou de ses suivantes.

	0348	20 0001 0371	(0001) = 60 yyyy 0100 avec "yyyy" égal à l'adresse, augmentée de l'unité, du dernier terme à diviser.
	0371	16 0093 0126	(2) (0093) = 00 (p) 0000 (8002) = 60 xxxx 0100 avec "xxxx" égal à l'adresse du premier second membre de l'équation pivot. Avant d'effectuer la première division, nous allons modifier, en (2), les contenus de 0007 et 0037.

Le pivot, pris dans la k° équation, est nul.
Les (n - k) dernières équations n'ont pas encore fourni de pivot; nous allons y rechercher un pivot non nul pour éliminer la k° inconnue.
L'adresse du terme de rang k dans une équation quelconque, se déduit de celle du terme de même rang dans l'équation qui la précède, en lui ajoutant (n + p).

	0314	65 0070 0277	(0070) = 00 xxxx 0000 avec "xxxx" égal à l'adresse du terme de rang k, dans la k° équation.
	0277	15 0130 0235	(0130) = 60 0000 0322
(3)	0235	15 0042 0122	(0042) = 00 (n + p) 0000
	0122	21 0028 0331	(0028) = 00 (d) 0000 avec "d", initialement égal à zéro, représentant le nombre d'équations déjà examinées parmi les (n - k) dernières équations.

```
0331    20 0037 0172        (0037) = 60 yyyy 0322
                            avec "yyyy" égal à l'adresse
                            du terme de rang k, dans la
                            (k + d + 1)° équation.
0172    10 0010 0222        (0010) = 00 (  k  ) 0000
0222    11 0036 0272        (0036) = 00 (  n  ) 0000
0272    44 0037 0327
```

Tous les termes de rang k,dans les (n - k) dernières équations, ont été examinés. Nous n'avons pas trouvé de pivot non nul. La matrice du système linéaire est singulière, nous ne pouvons pas poursuivre les calculs.

```
0327    01 0000 9999
```

Tous les termes de rang k, dans les (n - k) dernières équations, n'ont pas été examinés.

```
0037    60 xxxx 0322        (8003) = valeur exacte du terme
                            de rang k dans la (k + d + 1)°
                            équation.
0322    44 0377 0228
```

Le terme de rang k,dans la (k + d + 1)° équation, est nul. Nous allons examiner le terme de même rang dans l'équation suivante.

```
0228    60 0028 0334        (0028) = 00 (  d  ) 0000
0334    10 0018 0278        (0018) = 00 0001 0000
0278    15 0037 0235  (3)   (0037) = 60 yyyy 0322
                            avec le "yyyy" défini au haut
                            de cette page.
```

Le terme de rang k, dans la (k + d + 1)° équation, est non nul. Ce sera le pivot pour l'élimination de la k° inconnue. Il doit être placé sur la diagonale principale de la matrice du système linéaire. Pour cela il faut permuter les deux enregistrements des k° et (k + d + 1)° équations. Chaque permutation portera sur (n + p) termes.

Préparation de la première permutation portant sur les premiers enregistrements des k° et (k + d + 1)° équations.

```
0377    11 8001 0285
0285    69 0388 0372        (8001) = 60 0142 0337
                            (8002) = 00 0000 0000
                            (8003) = 00 0000 0000
```

Début de la permutation.

```
(4) 0372    24 0001 0328    (0001) = 60 0142 0337
                            pour la permutation des
                            premiers enregistrements.
                                   = 65 0070 0026
                            pour la permutation des
                            deuxièmes enregistrements.
    0328    11 0010 0378    (0010) = 00 ( k ) 0000
    0378    10 0018 0279    (0018) = 00 0001 0000
    0279    21 0007 0329    (0007) = 00 (1 - k) 0000
                            pour la permutation des
                            premiers enregistrements.
                                   = 00 (n²+pn+1-k) 0000
                            pour la permutation des
                            deuxièmes enregistrements.
    0329    20 0028 0381    (0028) = 00 0000 0000
                            Remise à zéro du compteur
                            du nombre de termes permutés.
(5) 0381    65 0070 0379    (0070) = 00 xxxx 0000
                            avec "xxxx" égal à l'adresse
                            du terme de rang k dans la
                            k° équation.
    0379    15 0007 0384    (0007) = voir ci-dessus.
                            (8002) = 00 zzzz 0000
                            avec "zzzz" égal à l'adresse
                            du premier terme du premier
                            ou du deuxième enregistrement
                            de la k° équation.
    0384    15 0028 0180    (0028)
```

```
0180    69 0335 0230        (0335) = 65 0000 0236
0230    22 0335 0280        (0335) = 65 aaaa 0236
0280    69 0385 0330        (0385) = 24 0000 0286
0330    22 0385 0380        (0385) = 24 aaaa 0286
0380    65 0037 0132        (0037) = 60 yyyy 0322
                            avec "yyyy" égal à l'adresse
                            du terme de rang k dans la
                            (k + d + 1)e équation.
0132    15 0007 0186        (0007) = voir à la page
                                     précédente.
                            (8002) = 00 vvvv 0000
                            avec "vvvv" égal à l'adresse
                            du premier terme du premier
                            ou du deuxième enregistrement
                            de la (k + d + 1)e équation.
0186    15 0028 0182        (0028)
0182    69 0236 0232        (0236) = 69 0000 0385
0232    22 0236 0282        (0236) = 69 bbbb 0385
0282    69 0286 0332        (0286) = 20 0000 0382
0332    22 0286 0335        (0286) = 20 bbbb 0382
0335    65 aaaa 0236
0236    69 bbbb 0385
0385    24 aaaa 0286
0286    20 bbbb 0382
0382    60 0028 0336        (0028) = compteur du nombre
                            de termes permutés.
0336    10 0018 0386        (0018) = 00 0001 0000
0386    21 0028 0237
0237    11 0042 0287        (0042) = 00 (n + p) 0000
0287    44 0381 0001
```

La permutation en cours n'est pas terminée.

```
0381    Nous passons à la permutation du terme suivant.
        Voir en (5)
```

La permutation en cours est terminée.

Premier cas: Il s'agit de la permutation des premiers enregistrements.
Nous préparons la permutation des seconds enregistrements.

```
0001    60 0142 0337        (0142) = 00 (n² + pn) 0000
0337    69 0146 0372 (4)    (8001) = 65 0070 0026
                            (8002) = 00 0000 0000
                            (8003) = 00 (n² + pn) 0000
```

Deuxième cas: Il s'agit de la permutation des deuxièmes enregistrements.
Nous sommes prêts à diviser l'équation pivot par le pivot.

```
0001    65 0070 0026        Il s'agit de la première
                            instruction du Sous-Programme
                            de Division de l'Equation
                            Pivot par le Pivot.
```

SOUS PROGRAMME d'ELIMINATION

But du Sous-Programme

1°) cas de la première résolution

L'équation pivot étant la k° équation, qui vient d'être divisée par le pivot, il s'agit d'éliminer la k° inconnue dans les (n - 1) autres équations du système linéaire. Cette élimination se fera dans l'une quelconque de ces équations en lui soustrayant l'équation pivot multipliée par son coefficient de la k° inconnue (multiplicateur).

Au début de la première résolution du système linéaire, les n équations sont enregistrées les unes à la suite des autres, dans des mémoires consécutives à partir de l'adresse M. Chaque équation est composée de (n + p) termes.

Le pivot étant pris sur la diagonale principale de la matrice du système linéaire, les inconnues sont éliminées dans l'ordre. Au cours de chaque élimination, tous les nouveaux coefficients calculés sont mis en mémoire à la place de ceux dont ils sont issus; de sorte que les coefficients ne subissant aucun calcul ne sont pas détruits. En opérant de cette façon, au début de la k° élimination, chaque équation se compose de:

- (k - 1) premiers termes, ou multiplicateurs, qui ont été utilisés lors de l'élimination des (k - 1) premières inconnues, dans cette équation. Toutefois si le rang "d" de l'équation considérée est inférieur à "k", cette équation a servi d'équation pivot pour l'élimination de la d° inconnue, et son terme de rang "d" n'est pas un multiplicateur mais un pivot.
- ensuite, de (n-k+1) termes qui sont les coefficients des inconnues restant à éliminer. Le premier de ces coefficients est celui de la k° inconnue, et, exception faite pour l'équation pivot, il sera le multiplicateur à utiliser pour éliminer la k° inconnue dans cette équation.
- enfin, des p seconds membres.

Les calculs de la k° élimination ne portent que sur les termes du premier membre ayant un rang supérieur à k, et sur les seconds membres de toutes les équations sauf la k°, qui est l'équation pivot. A la fin de la k° élimination, la situation sera identique à ce qu'elle était au début, mais avec k = k + 1.

2°) cas de la deuxième résolution et de ses suivantes.

Par rapport à la première résolution, seuls les seconds membres du système linéaire ont été changés. Tous les calculs relatifs aux premiers membres des équations restent donc les mêmes que précédemment.

Au cours de la première résolution du système linéaire, nous avons conservé en mémoire les "n" séries de (n - 1) multiplicateurs associés aux "n" éliminations afin d'éviter tout nouveau calcul avec les premiers membres des "n" équations. De cette façon, les calculs de chacune des "n" éliminations ne porteront que sur les seconds membres de toutes les équations, sauf l'équation pivot. Pour chaque élimination, par exemple la k°, nous utiliserons la série de multiplicateurs enregistrés dans les mémoires initialement occupées par les coefficients de l'inconnue éliminée, c'est à dire la k° inconnue.

Entrée dans le Sous-Programme

Les contenus de 8001, 8002 et 8003 pouvant être quelconques, seules quatre mémoires devront être convenablement chargées.

(0010) = 00 (k) 0000
(0043) = 00 (n + p - k) 0000
(0070) = 00 [M + (k - 1)(n + p + 1)] 0000
en position facteur, nous avons l'adresse du pivot pour la k° élimination.
(0154) = Mot d'Orientation
= (0300) = 65 0035 0039
lors de la première résolution.

= (0270) = 16 0093 0320
à partir de la deuxième résolution.

D'autre part, il sera fait appel à un certain nombre de constantes chargées au tout début du Programme de Liaison.

(0036) = 00 (n) 0000
(0042) = 00 (n + p) 0000
(0093) = 00 (p) 0000
(0103) = 00 (M) 0000

Sortie du Sous-Programme

Elle est fixe. Nous quittons le Sous-Programme d'Elimination pour retourner au Programme de Liaison.

	0250	65 0103 0057	(0103) = 00 (M) 0000
	0057	15 0010 0015	(0010) = 00 (k) 0000 (8002) = 00 (M + k) 0000 en position facteur, nous avons l'adresse, augmentée de l'unité, du terme de rang k dans le premier membre de la première équation.
(6)	0015	10 0018 0023	(0018) = 00 0001 0000
	0023	21 0028 0031	(0028) = 00 (k_t) 0000 avec "k_t" représentant le numéro d'ordre de l'équation en cours de traitement. Initialement: k_t = 1.
	0031	20 0035 0038	(0035) = 00 xxxx 0000 avec "xxxx" égal à l'adresse du terme de rang (k + 1) dans le premier membre de la k_t^e équation.
	0038	11 0010 0065	(0010) = 00 (k) 0000
	0065	44 0019 0020	

L'équation que nous nous préparons à traiter n'est pas l'équation pivot; nous devons donc y éliminer la k° inconnue.

0019	10 8001 0027	
0027	15 0030 8002	(0030) = 68 9999 0153 (8002) = 69 yyyy 0153 avec yyyy = xxxx - 1 ; il s'agit de l'adresse du terme de rang k dans le premier membre de la k_t^e équation.
8002	69 yyyy 0153	
0153	24 0014 0067	(0014) = multiplicateur associé à la k_t^e équation lors de la k° élimination.
0067	65 0070 0025	(0070) = 00 zzzz 0000 avec: zzzz = M+(k-1)(n+p+1), c'est à dire l'adresse du pivot.
0025	15 0078 0033	(0078) = 61 0001 0050
0033	20 0037 0040	(0037) = 61 vvvv 0050 avec "vvvv" égal à l'adresse du terme de rang (k + 1) dans le premier membre de l'équation pivot.
0040	15 0043 0047	(0043) = 00 (n + p -k) 0000
0047	20 0001 0154	(0001) = 61 wwww 0050 avec "wwww" égal à l'adresse, augmentée de l'unité, du dernier second membre de l'équation pivot. (0154) = Mot d'Orientation

Premier cas: Il s'agit de la première résolution du système linéaire.

```
0154    65 0035 0039        (0035) = 00 xxxx 0000
                            avec "xxxx" égal à l'adresse
                            du terme de rang (k + 1) dans
                            le premier membre de la
                            k_t° équation.
(7) 0039    10 8002 0097
    0097    15 0350 0055    (0350) = 21 0000 0203
    0055    10 0150 0105    (0150) = 69 0000 0003
(8) 0105    21 0007 0060    (0007) = 69 qqqq 0003
                            instruction d'appel des
                            coefficients de la k_t°
                            équation.
    0060    20 0064 0037    (0064) = 21 qqqq 0203
                            instruction pour le stockage
                            des nouveaux coefficients de
                            la k_t° équation.
    0037    61 vvvv 0050
```

Séquence d'Entrée dans l'A.V.F. pour effectuer une multiplication.

```
0050    69 1996 0049        VMU
0049    24 1808 0011
0011    69 0014 0017        (0014) = multiplicateur
0017    24 1801 0004
0004    69 0007 1800        (0007) = instruction de sortie
```

```
1800
Sous-Programme
A. V. F.
1803        (1803) = (0007)
```

Séquence d'Entrée dans l'A.V.F. pour effectuer une addition.

```
1803    69 qqqq 0003
0003    24 1801 0054
0054    69 1902 0005        VAA
0005    24 1808 0061
0061    69 0064 1800        (0064) = instruction de sortie
```

```
1800
Sous-Programme
A. V. F.
1803        (1803) = (0064)
```

Mise en mémoire du résultat.

```
1803    21 qqqq 0203
0203    60 0037 0041        (0037) = 61 vvvv 0050
0041    10 0018 0073        (0018) = 00 0001 0000
0073    21 0037 0090
0090    11 0001 0155        (0001) = 61 wwww 0050
                            avec "wwww" égal à l'adresse,
                            augmentée de l'unité du
                            dernier second membre de
                            l'équation pivot.
0155    44 0009 0020
```

Tous les nouveaux coefficients de la k_t° équation n'ont pas été calculés.
Nous préparons le calcul du coefficient suivant.

```
0009    60 0007 0161        (0007) = 69 qqqq 0003
0161    15 0064 0069        (0064) = 21 qqqq 0203
0069    10 0018 0123        (0018) = 00 0001 0000
0123    15 8001 0105 (8)
```

Le traitement de la k_t° équation est terminée,
ou l'équation que nous nous préparons à traiter est l'équation pivot. ($k_t = k$).

```
0020    60 0028 0083        (0028) = 00 ( k_t ) 0000
0083    11 0036 0091        (0036) = 00 (  n  ) 0000
0091    44 0045 0046
```

Les "n" équations du système linéaire n'ont pas été toutes traitées ($k_t < n$).
Nous préparons le passage à l'équation suivante.

0045 10 8001 0253

0253 15 0035 0089 (0035) = 00 xxxx 0000
avec "xxxx" égal à l'adresse du terme de rang (k + 1) dans le premier membre de la k_t° équation.

0089 15 0042 0015 (6) (0042) = 00 (n + p) 0000
(8003) = 00 (k_t) 0000
(8002) = 00 xxxx 0000
avec "xxxx" égal à l'adresse du terme de rang (k + 1) dans le premier membre de la (k_t + 1)° équation.

Les "n" équations du système linéaire ont subi la k° élimination.

0046 Nous sortons du Sous-Programme d'Elimination pour retourner au Programme de Liaison.

Deuxième cas: Il s'agit de la deuxième résolution ou de l'une de ses suivantes.
Les calculs d'élimination n'intéressent que les seconds membres des équations.

0154 16 0093 0320 (0093) = 00 (p) 0000

0320 20 0037 0370 (0037) = 61 vvvv 0050
avec "vvvv" égal à l'adresse du premier second membre de l'équation pivot.

0370 65 0035 0271 (0035) = 00 xxxx 0000
avec "xxxx" égal à l'adresse du terme de rang (k + 1) dans le premier membre de la k_t° équation.

0271 15 0036 0321 (0036) = 00 (n) 0000

0321 16 0010 0039 (7) (0010) = 00 (k) 0000
(8002) = 00 rrrr 0000
avec "rrrr" égal à l'adresse du premier second membre de la k_t équation.

SOUS PROGRAMME de CALCUL des APPROXIMATIONS

But du Sous-Programme

A la fin de chaque résolution de système linéaire, les solutions obtenues occupent les mémoires initialement réservées aux séries de seconds membres correspondantes. Ainsi le i^e vecteur solution occupe les mémoires:

$M + n + i - 1$
$M + n + p + n + i - 1$
$M + 2(n + p) + n + i - 1$
.....
$M + (n - 1)(n + p) + n + i - 1$

D'autre part, nous avons prévu "pn" mémoires consécutives pour le stockage des approximations successives des vecteurs solutions associés aux "p" séries de seconds membres données. Ces "pn" mémoires sont réservées à la suite du double enregistrement des données, c'est à dire à partir de l'adresse: $[M + 2n(n + p)]$. De cette façon, nous stockerons les approximations successives du vecteur solution associé à la i^e série de seconds membres dans "n" mémoires consécutives à partir de l'adresse: $[M + 2n(n + p) + (i - 1)n]$.

Le but du Sous-Programme de Calcul des Approximations variera en fonction de l'avancement des calculs; deux cas sont à considérer.

Premier cas: Nous venons de terminer la première résolution du système linéaire.

Les solutions obtenues constituent la première approximation des "p" vecteurs solutions associés aux séries de seconds membres données. Aucun calcul n'est nécessaire, il suffit de transférer les "p" vecteurs solutions dans les mémoires réservées à cet effet à la suite du double enregistrement des données (voir ci-dessus) .

Deuxième cas: Nous venons de terminer la seconde résolution du système linéaire, ou l'une de ses suivantes.

Les solutions obtenues constituent les corrections à apporter à la dernière approximation calculée, afin d'obtenir une nouvelle approximation qui sera conservée en mémoire, à la place de l'ancienne. Dans ce but, il faut additionner aux "n" composantes de chacun des "p" vecteurs solutions, la composante de même rang du vecteur correction correspondant. Nous effectuerons ce travail en corrigeant d'abord les composantes de rang 1, des "p" vecteurs solutions; puis celles de rang 2; et ainsi de suite jusqu'au rang "n".

Parallèlement aux calculs précédents, nous comparerons les résultats des deux dernières approximations de façon à arrêter les calculs dès que toutes les corrections seront sans influence sur les résultats. Dans cette dernière hypothèse, nous rechercherons la correction maximum en valeur absolue.

Entrée dans le Sous-Programme.

Les contenus de 8001, 8002 et 8003 pouvant être quelconques, seule la mémoire 0044 devra être convenablement chargée.

(0044) = 60 0000 0014
si l'on vient de terminer la première résolution du système linéaire.
= 60 0000 0305
si l'on vient de terminer la deuxième résolution du système linéaire, ou l'une de ses suivantes.

D'autre part, il sera fait appel à un certain nombre de constantes chargées au tout début du Programme de Liaison.

(0036) = 00 (n) 0000
(0093) = 00 (p) 0000
(0103) = 00 (M) 0000
(0157) = 00 (M + 2(n^2 + pn)) 0000

Sortie du Sous-Programme

Elle est fixe. Nous sortons du Sous-Programme de Calcul des Approximations, pour retourner au Programme de Liaison.

	0096	69 0149 0002	(0149) = 01 0777 9999
	0002	24 1843 0196	(1843)
			Nous modifions le contenu d'une mémoire du Sous-Programme A.V.F., afin d'être sûr de ne jamais avoir une correction, du même ordre de grandeur que la quantité qu'elle corrige.
	0196	69 0199 0052	(0199) = 11 8001 0207
	0052	24 0064 0117	(0064)
	0117	60 0103 0107	(0103) = 00 (M) 0000
	0107	20 0211 0164	(0211) = 00 0000 0000
	0164	24 0028 0081	(0028)
			Remise à zéro du compteur des quantités non affectées par les corrections.
(9)	0081	20 0085 0088	(0085) = 00 (k_c-1) 0000 avec "k_c" représentant le rang des composantes des vecteurs solutions que nous nous préparons à corriger. Nous avons initialement: $k_c = 1$
	0088	10 0036 0191	(0036) = 00 (n) 0000
	0191	10 0044 0249	(0044) = 60 0000 0014 si l'on vient de terminer la première résolution du système linéaire. = 60 0000 0305 à partir de la deuxième résolution.
	0249	21 0037 0140	(0037) = 60 xxxx 0014 avec "xxxx" égal à l'adresse de la composante de rang k_c, du premier vecteur solution. ou = 60 xxxx 0305 avec "xxxx" égal à l'adresse de la composante de rang k_c, du premier vecteur correction.
	0140	10 0093 0197	(0093) = 00 (p) 0000
	0197	21 0001 0204	(0001) = 60 yyyy 0014 avec "yyyy" égal à l'adresse de la composante de rang k_c, du dernier vecteur solution. ou = 60 yyyy 0305 avec "yyyy" égal à l'adresse de la composante de rang k_c, du dernier vecteur correction.
	0204	15 0157 0261	(0157) = 00 (M+2(n^2+pn)) 0000
	0261	15 0150 0255	(0150) = 69 0000 0003 (8002) = 69 zzzz 0003 avec "zzzz" égal à l'adresse de la composante de rang k_c du premier vecteur solution appartenant à la dernière approximation calculée.
(10)	0255	20 0007 0110	(0007) = 69 zzzz 0003 avec "zzzz" égal à l'adresse de la composante de rang k_c, d'un vecteur solution appar - tenant à la dernière approxi - mation calculée.
	0110	69 0063 0016	(0063) = 21 0000 0353
	0016	22 0014 0037	(0014) = 21 zzzz 0353 avec "zzzz" égal à l'adresse de la mémoire où doit être stockée la nouvelle compo -

sante de rang k_c, ...

Premier cas: Nous venons de terminer la première résolution du système linéaire.

Aucun calcul n'est nécessaire; il suffit de transférer le contenu de la mémoire d'adresse xxxx dans celle d'adresse zzzz.

```
0037    60 xxxx 0014
0014    21 zzzz 0353
0353    Voir à la suite de l'étude du Deuxième Cas, en (11).
```

Deuxième cas: Nous venons de terminer la seconde résolution du système linéaire, ou l'une de ses suivantes.

Il faut additionner le contenu de la mémoire d'adresse zzzz à celui de la mémoire d'adresse xxxx, et transférer le résultat dans la mémoire d'adresse zzzz.

```
0037    60 xxxx 0305
0305    24 0035 0007        (0035) = (xxxx)
```

Séquence d'Entrée dans l'A.V.F. pour effectuer une addition.

```
0007    69 zzzz 0003
0003    24 1801 0054
0054    69 1902 0005        VAA
0005    24 1808 0061
0061    69 0064 1800        (0064) = (0199)
                                   = instruction de sortie

1800
   Sous-Programme
     A. V. F.
                1803        (1803) = (0064)
```

REMARQUE: A la sortie du Sous-Programme A.V.F., les contenus de 8001, 8002 et 8003 sont identiques à ceux obtenus après l'éxécution de l'instruction équivalente traitant les nombres en Virgule Flottante Automatique.

Ainsi après une addition, nous trouvons dans 8001 le facteur de l'opération affecté de l'exposant le plus grand. Dans le cas présent, nous aurons: (8001) = (zzzz). Si ceci ne pouvait être, nous aurions l'arrêt: 01 0777 9999 (voir au tout début de la liste du Sous-Programme de Calcul des Approximations).

```
1803    11 8001 0207
0207    44 0311 0012
```

La composante de rang k_c d'un vecteur solution (contenu de la mémoire d'adresse zzzz) est affecté par la dernière correction calculée (contenu de la mémoire d'adresse xxxx). Nous stockons la nouvelle valeur de cette composante dans la mémoire d'adresse zzzz.

```
0311    10 8001 0014
0014    21 zzzz 0353
0353    Voir ci-dessous en (11)
```

La composante de rang k_c d'un vecteur solution n'est pas affectée par la dernière correction calculée.

```
0012    60 0028 0133        (0028) = comptage des compo -
                                     santes non affectées par les
                                     dernières corrections
                                     calculées.
0133    10 0018 0323        (0018) = 00 0001 0000
0323    21 0028 0131
0131    67 0211 0215        (8002) = correction maximum
                                     en valeur absolue
0215    18 0035 0139        (0035) = valeur exacte d'une
                                     correction.
0139    46 0092 0353
```

La dernière correction considérée est maximum en valeur absolue.

```
0092    24 0211 0353
```

Fin de l'étude du "Deuxième Cas".
Les instructions qui suivent sont communes aux deux cas.
Dans le premier cas, nous préparons le transfert de la composante de rang k_c du vecteur solution suivant.
Dans le second cas, nous préparons la correction de la composante de rang k_c du vecteur solution suivant.

```
(11)  0353    60 0037 0241        (0037) = 60 xxxx 0014
                                           ou
                                   60 xxxx 0305
      0241    10 0018 0373        (0018) = 00 0001 0000
      0373    21 0037 0190
      0190    11 0001 0355        (0001) = 60 yyyy 0014
                                           ou
                                   60 yyyy 0305
                                  avec "yyyy" égal à l'adresse
                                  de la dernière composante de
                                  rang k_c à considérer.
      0355    44 0059 0160
```

Toutes les composantes de rang k_c des p vecteurs disponibles n'ont pas été utilisées.

```
      0059    65 0007 0361        (0007) = 69 zzzz 0003
      0361    15 0036 0255  (10)   (0036) = 00 ( n ) 0000
```

Toutes les composantes de rang k_c des p vecteurs disponibles ont été utilisées.

```
      0160    65 0085 0189        (0085) = 00 (k_c - 1) 0000
      0189    15 0018 0024        (0018) = 00 0001 0000
      0024    16 0036 0291        (0036) = 00 (  n  ) 0000
      0291    45 0094 0145
```

Les calculs que nous venons de terminer ne portaient pas sur les composantes de rang n.
Nous préparons une nouvelle série de calcul portant sur les composantes de rang $(k_c + 1)$.

```
      0094    15 8001 0254
      0254    10 0001 0006        (0001) : voir plus haut
      0006    11 0044 0081  (9)   (0044) = 60 0000 0014
                                           ou
                                   60 0000 0305
                                  (8003) = 00 yyyy 0000
                                  avec "yyyy" égal: soit à
                                  l'adresse de la dernière
                                  composante de rang k_c ;
                                  soit à l'adresse, diminuée
                                  de "n", de la première
                                  composante de rang (k_c + 1).
```

Les calculs que nous venons de terminer portaient sur les composantes de rang n.
Nous venons de calculer une nouvelle approximation des vecteurs solutions associés aux "p" séries de seconds membres données.

```
      0145    69 0144 0247        (0144) = 69 1846 1849
      0247    24 1843 0246        (1843): Nous rétablissons le
                                  contenu normal de cette
                                  mémoire qui avait été modifié
                                  au tout début de ce Sous -
                                  Programme.
      0246    Nous sortons du Sous-Programme de Calcul des
              Approximations pour retourner au Programme de
              Liaison.
```

SOUS PROGRAMME de CALCUL des RESIDUS

But du Sous-Programme

Il s'agit de calculer les "n" composantes de chacun des "p" vecteurs résidus correspondants à la dernière approximation des vecteurs solutions associées aux "p" séries de seconds membres données.

Pour ces calculs, nous utilisons le deuxième enregistrement du système linéaire, rangé ligne par ligne, qui occupe $(n^2 + pn)$ mémoires consécutives à partir de l'adresse $[M + n^2 + pn]$. Chaque ligne se compose des "n" coefficient du premier membre et des "p" seconds membres, c'est à dire d'une équation complète du système.

Les "p" vecteurs solutions sont conservés, les uns à la suite des autres, dans "pn" mémoires consécutives à partir de l'adresse $[M + 2(n^2 + pn)]$. De cette façon, le vecteur solution associé à la i° série de seconds membres occupe "n" mémoires consécutives à partir de l'adresse $[M + 2n(n + p) + (i - 1)n]$.

Les vecteurs résidus seront mis à la place initialement occupée par le premier enregistrement des "p" vecteurs seconds membres données. Ainsi, le vecteur résidu associé au i° vecteur solution sera stocké dans les mémoires d'adresses:

$M + n + i - 1$
$M + n + p + n + i - 1$
$M + 2(n + p) + n + i - 1$
.....
$M + (n - 1)(n + p) + n + i - 1$

Le calcul des résidus présente par ailleurs les particularité suivantes:

- chaque vecteur résidu est égal à

  $b - Ax$

  si l'on appelle: b le vecteur second membre, x le vecteur solution et A la matrice du système linéaire.

- les calculs sont effectués en Virgule Flottante DOUBLE PRECISION à partir de données en simple précision. Les résultats conservés en mémoire seront exprimés en Virgule Flottante Simple Precision.

  Pour réaliser celà, nous disposons d'un Sous-Programme d'Addition en Virgule Flottante DOUBLE PRECISION et nous modifierons le Sous-Programme A.V.F. de façon à obtenir en DOUBLE PRECISION le produit de deux nombres exprimés en Virgule Flottant Simple Précision.

Entrée dans le Sous-Programme

Les contenus de 8001, 8002 et 8003 pouvant être quelconques, il sera seulement fait appel à un certain nombre de constantes chargées au tout début du Programme de Liaison.

(0036) = 00 (n) 0000
(0042) = 00 (n + p) 0000
(0093) = 00 (p) 0000
(0103) = 00 (M) 0000
(0142) = 00 (n^2+pn) 0000
(0157) = 00 (M + 2(n^2 + pn)) 0000

Sortie du Sous-Programme

Elle est sous la dépendance du Programme de Liaison par l'intermédiaire de la mémoire de travail 0365.

	0299	65 0102 0257	(0102) = 36 0000 0125
	0257	24 1912 0265	(1912) Nous modifions le contenu de cette mémoire du Sous-Programme A.V.F. de façon à obtenir en DOUBLE PRECISION le produit de deux nombres exprimés en Virgule Flottante Simple Précision.
(12)	0265	21 0028 0181	(0028) = 00 (i - 1) 0000 avec "i" représentant le numéro d'ordre du vecteur résidu en cours de calcul. Nous aurons: $1 \leqslant i \leqslant p$
	0181	10 0103 0307	(0103) = 00 (M) 0000
	0307	10 0036 0391	(0036) = 00 (n) 0000
	0391	10 0194 0349	(0194) = 21 0000 0354
	0349	21 0064 0167	(0064) = 21 xxxx 0354 Instruction utilisée pour la mise en mémoire des n compo - santes du i° vecteur résidu. Initialement: xxxx = M + n + i - 1
	0167	65 0103 0357	(0103)
	0357	21 0085 0138	(0085) = 00 (k - 1) 0000 avec "k" représentant le rang de la composante, en cours de calcul, appartenant au i° vecteur résidu. Nous aurons: $1 \leqslant k \leqslant n$.
	0138	15 0142 0297	(0142) = 00 (n^2 + pn) 0000
	0297	15 0101 0056	(0101) = 61 0000 0007
(13)	0056	20 0037 0240	(0037) = 61 yyyy 0007 Instruction utilisée pour appeler tous les coefficients du premier membre de la k° équation. Initialement, k = 1 et yyyy = M + n^2 + pn.

0240	15 0036 0192	(0036) = 00 (n) 0000
0192	20 0001 0304	(0001) = 61 zzzz 0007 avec "zzzz" égal à l'adresse, augmentée de l'unité, du dernier coefficient du premier membre de la k° équation. Initialement, k = 1 et zzzz = M + n^2 + pn + n .
0304	15 0028 0233	(0028) = 00 (i - 1) 0000
0233	69 0136 0239	(0136) = 65 0000 0106
0239	22 0136 8001	(0136) = 65 vvvv 0106 avec "vvvv" égal à l'adresse du i° second membre de la k° équation.
8001	65 vvvv 0106	
0106	35 0002 0113	
0113	11 8003 0021	
0021	24 0074 0077	(0074) Cette mémoire contient, en position instruction suivante, l'exposant du i° second membre de la k° équation.
0077	35 0009 0347	
0347	21 0152 0156	(0152) = 0m mmmm mmm0 il s'agit des neuf premiers chiffres de la mantisse du second membre précédent
0156	20 0062 0315	(0062) = 00 0000 0000 il s'agit des dix chiffres suivants de la mantisse du second membre précédent.
0315	60 0028 0283	(0028) = 00 (i - 1) 0000
0283	19 0036 0008	(0036) = 00 (n) 0000
0008	30 0004 0119	
0119	15 0157 0112	(0157) = 00 (M+2(n^2+pn)) 0000
0112	15 0200 0206	(0200) = 69 0000 0053

```
0206    20 0007 0037        (0007) = 69 wwww 0053
                            avec "wwww" égal à l'adresse
                            de la mémoire contenant la
                            première composante du i°
                            vecteur solution.
(14) 0037   61 yyyy 0007
```

Séquence d'Entrée dans l'A.V.F. pour effectuer une multiplication.

```
0007    69 wwww 0053
0053    24 1801 0104
0104    69 1996 0099        VMU
0099    24 1808 0111
0111    69 0114 1800        (0114) = instruction de sortie
```

```
1800
   Sous-Programme
     A. V. F.
              1803          (1803) = (0114)
```

REMARQUE: A la suite de la modification précé - demment apportée au Sous-Programme A.V.F., nous disposons du produit, changé de signe, des contenus des mémoires d'adresse yyyy et wwww, en DOUBLE PRECISION de la façon suivante:

- en 1879, nous avons l'exposant du résultat en position instruction suivante.
- en 1890, nous avons les dix premiers chiffres de la mantisse.
- en 1831, nous avons les chiffres suivants de la mantisse.

A l'aide du Sous-Programme d'Addition en Virgule Flottante DOUBLE PRECISION, nous allons additonner au produit précédent: un nombre en DOUBLE PRECISION contenu dans les mémoires 0074, 0152 et 0062. (voir l'étude du Sous-Programme d'Addition en Virgule Flottante Double Précision à la suite de celle du Sous-Programme de Calcul des Résidus).

```
0114                        (0114) = (1803)
   Sous-Programme
   d'Addition en
 Virgule Flottante
  DOUBLE PRECISION
              0151
```

REMARQUE: Le résultat de l'addition se trouve, en virgule flottante double précision, dans les mémoires 0074, 0152 et 0062.

Nous préparons le produit suivant.

```
0151    60 0037 0242        (0037) = 61 yyyy 0007
0242    15 0007 0162        (0007) = 69 wwww 0053
0162    10 0018 0124        (0018) = 00 0001 0000
0124    15 8001 0333
0333    21 0037 0290
0290    20 0007 0210
0210    11 0001 0256        (0001) = 61 zzzz 0007
0256    44 0037 0260
```

Tous les coefficients du premier membre de la k° équation n'ont pas été utilisés. Le calcul de la composante de rang k du i° vecteur résidu n'est pas terminé.

```
0037    Voir en (14)
```

Tous les coefficients du premier membre de la k° équation ont été utilisé. Le calcul de la composante de rang k du i° vecteur résidu est terminé.
Cette composante est disponoble en virgule flottante double précision dans les mémoires 0074, 0152 et 0062. Avant de la mettre en mémoire, il faut la transformer en virgule flottante simple précision à l'aide d'une partie du Sous-Programme A.V.F.

```
0260    69 0064 0217        (0064) = 21 xxxx 0354
0217    24 1803 0358        (1803) = instruction de sortie
                            du Sous-Programme A.V.F.
```

0358	60 0152 0159	(0152) = 0r rrrr rrrr il s'agit des neuf premiers chiffres de la mantisse du nombre en virgule flottante double précision.
0159	15 0062 0367	(0062) = rr rrrr rrrr il s'agit des dix chiffres suivants de la même mantisse.
0367	69 0074 0227	(0074) = 00 0000 00RR avec "RR" représentant l'exposant du nombre en virgule flottante double précision.
0227	24 1879 0082	(1879) Mémoire de travail du Sous-Programme A.V.P.
0082	31 0009 1998	

1800 Sous-Programme	1998
1998 A. V. P.	1803

1803	21 xxxx 0354	(1803) = (0064) (8003) = RR rrrr rrrr Il s'agit de la composante de rang k du i° vecteur résidu, exprimée en virgule flottante simple précision.
0354	60 0085 0289	(0085) = 00 (k - 1) 0000
0289	10 0018 0174	(0018) = 00 0001 0000
0174	21 0085 0188	
0188	11 0036 0292	(0036) = 00 (n) 0000
0292	44 0195 0296	

Les "n" composantes du i° vecteur résidu n'ont pas toutes été calculées.
Nous préparons le calcul de la composante suivante.

0195	65 0064 0169		(0064) = 21 xxxx 0354 Instruction utilisée pour la mise en mémoire des "n" composantes du i° vecteur résidu.
0169	15 0042 0397		(0042) = 00 (n + p) 0000
0397	20 0064 0267		(0064) = 21 xxxx 0354 avec "xxxx" égal à l'adresse de la mémoire où sera stockée la (k + 1)° composante du i° vecteur résidu.
0267	65 0001 0306		(0001) = 61 zzzz 0007 avec "zzzz" égal à l'adresse, augmentée de l'unité, du dernier coefficient du premier membre de la k° équation.
0306	15 0093 0056	(13)	(0093) = 00 (p) 0000 (8002) = 61 yyyy 0007 avec "yyyy" égal à l'adresse du premier coefficient du premier membre de la (k + 1)° équation. Nous supposons maintenant: k = k + 1.

Les "n" composantes du i° vecteur résidu ont toutes été calculées.

0296	60 0028 0383	(0028) = 00 (i - 1) 0000
0383	10 0018 0224	(0018) = 00 0001 0000
0224	11 0093 0048	(0093) = 00 (p) 0000
0048	44 0201 0202	

Les "p" vecteurs résidus n'ont pas tous été calculés.
Nous préparons le calcul du vecteur résidu suivant.

0201 10 8001 0265 (12)

Les "p" vecteurs résidus ont tous été calculés.

0202 69 0356 0109 (0356) = 68 8001 1870

0109 24 1912 0365 (1912): Nous rétablissons le contenu normal de cette mémoire qui avait été modifié au tout début de ce Sous-Programme.

0365 Nous sortons du Sous-Programme de Calcul des Résidus pour retourner au Programme de Liaision.

SOUS PROGRAMME d'ADDITION en VIRGULE FLOTTANTE DOUBLE PRECISION

But du Sous-Programme

Il s'agit d'additionner deux nombres A et B exprimés en Virgule Flottante Double Précision.

Par nombre en Virgule Flottante Double Précision, nous entendons tout nombre N écrit sous la forme:

$$\underbrace{NN}_{E}\ \underbrace{nnnn.....nnnn}_{M}\ \underbrace{\pm}_{S}$$

avec: E: exposant (toujours deux chiffres pris avec le signe +)
M: mantisse (19 ou 20 chiffres)
S: signe du nombre tout entier.

et qui est égal à:

$$N = \pm\ 0,nnnn.....nnnn \times 10^{(NN - 50)}$$

Un tel nombre occupera trois mémoires:

- une première pour son exposant

00 0000 00NN $\pm$

- deux autres mémoires pour sa mantisse:

$\overset{0}{n}n$ nnnn nnnn S

nn nnnn nnnn S

Entrée dans le Sous-Programme

Le nombre A doit être chargé dans trois mémoires, de la façon suivante:

(0074) = 00 0000 00AA $\pm$
(0152) = 0a aaaa aaaa
(0062) = aa aaaa aaaa

De même, le nombre B sera chargé dans trois autres mémoires:

(1879) = 00 0000 00BB $\pm$

(1890) = bb bbbb bbbb
(1831) = bb bbbb bbbb

REMARQUE

La mantisse du nombre B n'aura jamais plus de 16 chiffres. En effet, ce nombre sera toujours le produit de deux nombres exprimés en virgule flottante simple précision.

Les contenus de 8001, 8002 et 8003 peuvent être quelconques. La première instruction du Sous-Programme se trouve en 0114.

Sortie du Sous-Programme.

Le résultat R occupe les mémoires initialement réservées au nombre A. Nous aurons, de la même façon que pour A:

(0074) = 00 0000 00RR $\pm$
(0152) = 0r rrrr rrrr
(0062) = rr rrrr rrrr

Nous sortons toujours du Sous-Programme d'Addition en Virgule Flottante DOUBLE PRECISION pour retourner dans le Sous-Programme de Calcul des Résidus.

0114	67 0074 0029		(0074) = 00 0000 00AA $\pm$
0029	21 0211 0214		(0211) = 00 0000 0000
0214	18 1879 0084		(1879) = 00 0000 00BB $\pm$
0084	35 0004 0245		(8002) = 00 (|AA| - |BB|) 0000
0245	46 0098 0399		

Ayant pris les valeurs absolues, l'exposant du nombre B est supérieur à celui du nombre A . Nous allons permuter les nombres A et B .

0098	24 0074 0127		(0074) = 00 0000 00BB $\pm$
0127	20 0070 0274		(0070) = 00 (|AA| - |BB|) 0000
0274	60 1890 0295		(1890) = bb bbbb bbbb
0295	15 1831 0135		(1831) = bb bbbb bbbb
0135	30 0001 0342		
0342	69 0152 0058		(0152) = 0a aaaa aaaa
0058	24 1890 0143		(1890) = 0a aaaa aaaa
0143	69 0062 0066		(0062) = aa aaaa aaaa
0066	24 1831 0134		(1831) = aa aaaa aaaa
0134	21 0152 0108		(0152) = 0b bbbb bbbb
0108	20 0062 0116		(0062) = bb bbbb bbbb
0116	67 0070 0175	(15)	(8002) = 00 (|AA| - |BB|) 0000

REMARQUE

Les mantisses des nombres A et B sont maintenant cadrées de la même façon dans les deux couples de mémoires:

- 1890 et 1831 pour A
- 0152 et 0062 pour B.

Par contre, les exposants restent inchangés.

Ayant pris les valeurs absolues, l'exposant du nombre A est supérieur ou égal à celui du nombre B.
Par conséquent, pour additionner les mantisses de ces deux nombres, nous décalerons celle de B par rapport à celle de A, de (|AA| - |BB|) positions vers la droite.
Par ailleurs, la mantisse de B n'est pas cadrée, dans les mémoires 1890 et 1831, de la même façon que celle de A, dans les mémoires 0152 et 0062. Pour palier à cette anomalie, nous allons augmenter le nombre de décalages de un .

0399	17 0018 0175	(15)	(0018) = 00 0001 0000
			(8002) = 00 (|AA|-|BB|+ 1) 0000

Que $(|AA| - |BB|)$ soit négatif, positif ou nul, l'addition des nombres A et B peut maintenant se faire de la même façon. Nous décalerons toujours la mantisse contenue dans les mémoires 1890 et 1831 par rapport à celle contenue dans les mémoires 0152 et 0062, de $\left[\left|\,|(0074)| - |(1879)|\,\right| + 0 \text{ ou } 1\right]$ positions vers la droite. L'exposant du résultat sera toujours

égal au contenu de 0074, mais il pourra être modifié en fonctions des opérations de recadrage de la mantisse du résultat.
Afin de conserver tout leur sens aux notations définies au début de cet exposé, nous allons supposer que |AA| est supérieur ou égal à |BB|. D'autre part, nous allons appeler "D" le nombre de décalages contenu en position facteur de 8002.

```
(15)  0175    18 0128 0184        (0128) = 00 0010 0000
      0184    46 0087 0238        (8002) = 00 (D - 10) 0000
```

Le nombre de décalages est inférieur à dix, et nous sommes sûrs qu'il n'est pas nul.

```
      0087    15 8001 0345
(16)  0345    69 0148 0251        (0148) = 31 0000 0043
      0251    22 0070 0324        (0070) = 31 000D 0043
      0324    69 0177 0080  (17)  (0177) = 00 0000 0198
```

Le nombre de décalages est supérieur ou égal à dix.

```
      0238    45 0392 0345
```

Le nombre de décalages est égal à dix.
Comme nous utilisons un code "DDA" (Décalage à Droite avec Arrondi), le cas D = 10 sera résolu de la même façon que le cas D < 10.

```
      0345    Voir en (15)
```

Le nombre de décalages est supérieur à dix.
Le décalage de D position à droite se fera en deux fois, à l'aide de deux codes "DDA".

```
      0392    69 0148 0301        (0148) = 31 0000 0043
      0301    24 0070 0374        (0070) = 31 0000 0043
                                  Décalage de dix positions à
                                  droite, avec arrondi.
                                  (8002) = 00 (D - 10) 0000
                                  Posons: D' = D - 10
      0374    18 0128 0234        (0128) = 00 0010 0000
      0234    45 0288 0339        (8002) = 00 (D' - 10) 0000
```

Le nombre de décalages est différent de vingt (D≠20 et D'≠10).

```
      0288    46 0193 0151
```

Le nombre de décalages est supérieur à vingt (D>20 et D'>10).
Le nombre B est négligeable devant le nombre A qui est pris comme résultat de l'addition.
Le nombre A occupe les mémoires 0074, 0152 et 0062 et sa mantisse a un cadrage qui verifie les conditions de "Sortie du Sous-Programme":

```
      0151    Nous sortons du Sous-Programme d'Addition en
              en Virgule Flottante DOUBLE PRECISION pour
              retourner au Sous-Programme de Calcul des
              Résidus.
```

Le nombre de décalages est inférieur à vingt (10<D<20 et D'<10)

```
      0193    10 8001 0339        (8002) = 00 ( D' ) 0000
(18)  0339    69 0243 0080  (17)  (0243) = 31 0000 0198
```

Le nombre de décalages est égal à vingt (D=20 et D'=10).
Comme nous utilisons un code "DDA" (Décalage à Droite avec Arrondi), le cas D'= 10 sera résolu de la même façon que le cas D'<10.

```
      0339    Voir en (18)

(17)  0080    22 0043 0346        (0043) = 00 000D 0198
                                           si D<10
                                        = 31 000D'0198
                                           si 10 < D <20
                                        = 31 0000 0198
                                           si D = 20
      0346    60 1890 0395        (1890) = bb bbbb bbbb
      0395    15 1831 0070        (1831) = bb bbbb bbbb
      0070    31 000D 0043        REMARQUE: Nous pouvons avoir
                                  D = 1, 2, ... , 9
                                  Si D = 10,
                                     (0070) = 31 0000 0043
```

```
0043    00 000D 0198        REMARQUE: Si D ⩽ 10
          ou
0043    31 000D'0198        Si  10 < D < 20
                            Si  D = 20
                                (0043) = 31 0000 0198
0198    10 0152 0158        (0152) = 0a aaaa aaaa
0158    15 0062 0317        (0062) = aa aaaa aaaa
0317    45 0120 0071
```

La mantisse du résultat n'est pas nulle.
Il faut maintenant cadrer cette mantisse avec un seul zéro à gauche dans 8003.

```
0120    44 0225 0275
```

Le contenu de 8003 est nul.

```
0275    69 0178 0231        (0178) = 00 0000 0010
0231    24 0211 0264        (0211) = 00 0000 0010
0264    60 8002 0225
```

Le contenu de 8003 était nul; nous venons de transférer le contenu de 8002 dans 8003, et nous avons prévu la conséquence de cette opération sur l'exposant du résultat (il faut diminuer l'exposant de 10).
ou, le contenu de 8003 n'est pas nul.

```
0225    36 0000 0248        (8003) = rr rrrr rrrr
                            (8002) = rr rrrr rrXX
                            avec "XX" représentant le
                            nombre de décalages vers la
                            gauche.
0248    20 0070 0325        (0070) = rr rrrr rrXX
0325    36 0000 0298        (8002) = rr rrrr rr00
0298    30 0001 0208
0208    21 0152 0258        (0152) = 0r rrrr rrrr
0258    20 0062 0166        (0062) = rr rrrr rrr0
```

Calcul de l'exposant du résultat.
Il est égal au contenu de la mémoire 0074 avant le recadrage de la mantisse du résultat. Cette dernière opération a nécessité la suppression des [(XX - 1) + 0 ou 10] zéros situés à gauche du premier chiffre significatif. Par conséquent, l'exposant du résultat sera égal à:

(0074) - [(XX - 1) + 0 ou 10]

```
0166    68 0070 0375        (0070) = rr rrrr rrXX
0375    35 0008 0293
0293    65 8002 0351
0351    30 0008 0219        (8002) = 00 0000 00XX -
0219    17 0074 0079        (0074) = 00 0000 00AA + -
0079    17 0032 0137        (0032) = 00 0000 0001
0137    18 0211 0216        (0211) = 00 0000 0000
                                        ou
                                    00 0000 0010
0216    46 0071 0170
```

L'exposant du résultat est positif ou nul.

```
0170    20 0074 0151        (0074) = 00 0000 00RR
(19) 0151   Nous sortons du Sous-Programme d'Addition en
            Virgule Flottante DOUBLE PRECISION pour retourner
            au Sous-Programme de Calcul des Résidus.
```

La mantisse du résultat est nulle,
ou l'exposant du résultat est négatif.
Dans ces deux cas, le résultat de l'addition est nul. Nous transférons des zéros dans les trois mémoires contenant ce résultat.

```
0071    21 0152 0308        (0152) = 00 0000 0000 + -
0308    24 0062 0266        (0062) = 00 0000 0000 + -
0266    24 0074 0151        (0074) = 00 0000 0000 + -
0151    Voir en (19)
```

I, 5 PROGRAMMES de RESOLUTION de SYSTEMES LINEAIRES en SIMPLE et en DOUBLE PRECISION. METHODE de JORDAN PROGRESSIVE (Références J S P 2 et J D P 2).

I, 5-1 NOTICES d'UTILISATION

I, 5-1-1 PROGRAMME en SIMPLE PRECISION

Objet : Etant donné en simple précision, la matrice et les vecteurs seconds membres d'un système linéaire, ce programme calcule en Virgule Flottante *Flair* et en Simple Précision, les vecteurs solutions.

Conditions d'Utilisation : Si «*n*» désigne l'ordre de la matrice du système linéaire et «p» le nombre de vecteurs seconds membres, la limite d'utilisation de ce programme est :

$$\frac{(n+p)\ (n+p+6)\ +1}{4} \leq 1449$$

Ainsi avec p = 1, nous avons : $n_{MAX} = 72$

La Virgule Flottante simulée étant du type *Flair*, on représentera :

1 par 51 1000 0000 +

Il faudra enregistrer l' A V F (Arithmétique en Virgule Flot tante) sur le tambour.

Présentation des données : Les données seront toujours précédées d'une *carte caractéristique* composée de la façon suivante :

Mot 1 — 00 (*n*) 0000 +
. «*n*» étant l'ordre du système linéaire à résoudre.

Mot 2 — 00 (p) 0000 +
. «p» étant le nombre de séries de seconds membres.

Mot 3 — La valeur minimale tolérée pour le pivot, exprimée en virgule flottante *Flair* avec le signe + :

Mot 4 à 8 — 00 0000 0000 $\pm$

cette carte doit être «non - chargement».

Le système linéaire à résoudre sera perforé équation par équation, (chaque équation comprenant les premier et seconds membres, (*n* + p) coefficients par équation). Le modèle de carte utilisé sera le suivant :

Mot 1 NE NC 0 0 0 0 0 NB +

avec NE : numéro de l'équation, nous aurons : $1 \leq NE \leq n$

NC : numéro de carte de l'équation, nous aurons : $1 \leq NC \leq \frac{n+p}{7} + (0 \text{ ou } 1)$

NB : nombre de mots de la carte à enregistrer : $1 \leq NB \leq 7$

Mots 2 à 8 Ils contiennent les coefficients à enregistrer.

Toutes les cartes de ce modèle seront, elles aussi, «non chargement».

Forme du programme : Le programme, y compris l'A V F, se présente sous la forme «6 mots» par carte. Le mot 8 est utilisé pour le contrôle du chargement.

Le programme lui-même est précédé d'un programme de chargement (cartes jaunes) et immédiatement suivi d'une carte «départ calcul automatique», faisant débuter leprogramme en 1 651.

Encombrement du programme : Le programme propr ement dit occupe les mémoires 1 450 à 1 799. Les mémoires 1 800 à 1 999 sont occupéespar l'Arithmétique en Virgule Flottante (A V F).

Déroulement du programme : Les interrupteurs du pupitre seront positionnés de la façon suivante :

STOP sur «effectif»
PROGRAMME sur «normal»
CONTROLE sur «marche»
INDICATEURS sur «registre programme»
DEPASSEMENT DE CAPACITE et ERREURS sur «arrêt»

Le tableau de connexions de l'unité de lecture sera le tableau standard «8 mots».

On procèdera de la façon suivante :

Remplir le magasin de perforation et faire «Départ Perforation». Placer le programme dans le magasin de lecture, et immédiatement derrière : lacarte caractéristique et les carte s contenant les coefficients du système à résoudre. Afficher au pupitre :

70 1951 9999

Faire «Restauration Générale», «Départ Programme», et «Départ Lecture».

Une fois le programme enregistré, la lecture de la carte caractéristique et des données, ainsi que le départ des calculs se font sans intervention de l'opération.

Les calculs terminés, les résultats seront perforés et nousaurons au pupitre :

01 0999 1651

La machine est prête à traiter le système linéaire suivant, sans réenregistrement du programme.

Cartes «RESULTATS» : Chaque vecteur solution est perforé suivant le modèle de carte ci-dessous :

Mot 1 NS NC 0 0 0 0 0 NB +

avec : NS numéro d'ordre du vecteur solution : $1 \leq NS \leq p$

NC numéro de carte pour le vecteur solution : $1 \leq NC \leq \frac{n}{7} + (0 \text{ ou } 1)$

NB nombre de mots de la carte : $1 \leq NB \leq 7$

Mots 2 à 8 : Résultats.

Une fois tous les résultats perforés, une carte supplémentaire est obtenue, elle donne la perte maximale de précision. Cette carte est composée de la façon suivante :

Mot 1 : 00 00xx 0000 +

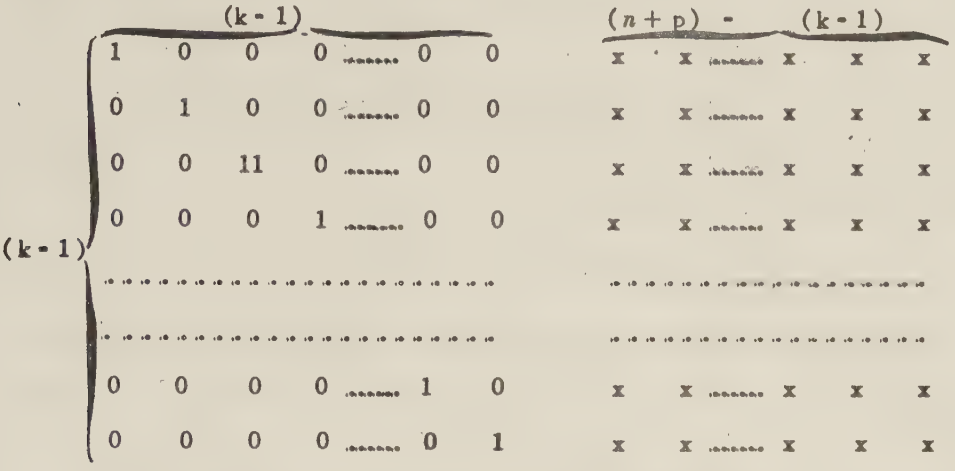
xx : élimination au cours de laquelle a eu lieu la perte de précision maximale.

Mot 2 : 00 0000 00yy +

yy : nombre maximal de chiffres perdus.

Mot 3 à 8 : 00 0000 0000 -

Méthode utilisée : Il s'agit de la méthode progressive JORDAN . C'est-à-dire que lors de la lecture de la k° équation, les (k - 1) équations précédentes ont subi (k - 1) éliminations . Le système présente alors la forme suivante :



La k° équation subira à son tour les (k - 1) éliminations et servira d'équation pivot pour la k° qui sera immédiatement effectuée sur les (k - 1) équations déjà traitées.

Pour toutes ces éliminations, le pivot est pris obligatoirement sur la diagonale principale.

Si l'on rencontre un pivot inférieur au minimum admis, l'équation en cause est perforée et devra être replacée derrière la n° équation, dans le magasin de lecture .

Si tous les pivots possibles sont inférieurs au minimum admis, nous avons l'arrêt :

01 9999 9999

Arrêts programmés :

01	0999	1651	Fin des calculs
01	9999	9999	Pivots tous inférieurs au minimum toléré .
01	0111	8000	Arrêt OVERFLOW . L'exposant du résultat d'une opération en arithmétique flottante est supérieur à 99 .
01	0222	8000	Lors d'une division, nous avons un diviseur nul .
01	0001	1530	La carte que vient de lire la machine n'est pas à sa place . Le contenu du distributeur nous indique alors les références de la carte que la machine désire lire . Le contenu de 8001 est : 00 0000 (NE) (NC) (NE) : numéro d'équation (NC) : numéro de carte .

Il suffit alors de vider l'unité de lecture et de remettre les cartes dans l'ordre convenable. En appuyant sur «Départ Programme» , la lecture reprendra normalement en 1530.

I, 5-1-2 PROGRAMME en DOUBLE PRECISION

Objet : Etant donné en DOUBLE PRECISION, la matrice et les vecteurs seconds membres d'un système linéaire, ce Programme calcule en Virgule Flottante *Flair* et en *Double Précision*, les vecteurs solutions.

Conditions d'Utilisation : Si «*n*» désigne l'ordre de la matrice du système linéaire et «p» le nombre de vecteurs seconds membres, la limite d'utilisation de ce Programme est :

$$\frac{(n+p)\ (n+p+6)\ +\ 1}{2} \leqslant 1299$$

Ainsi, avec p = 1 , nous avons $n_{MAX} = 47$

La virgule flottante étant du type *Flair*, on représentera :

1 par $\underbrace{51\quad 1000\quad 0000}_{\text{I}}\qquad \underbrace{00\quad 0000\quad 0000}_{\text{II}}$

Dans tout ce qui suit, nous appellerons I, la partie gauche et II, la partie droite d'un nombre en *Double Précision*.

Il faudra enregistrer l'A.D.P. (Arithmétique en virgule flottante Double Précision) sur le tambour.

Présentation des données : Les données seront toujours précédées d'une *Carte Caractéristique* composée de la façon suivante :

Mot 1 oo (*n*) oooo

«*n*» étant l'ordre du système linéaire à résoudre

Mot 2 oo (p) oooo

«p» étant le nombre de vecteurs seconds membres

Mot 3 La valeur minimum tolérée pour le pivot : exprimée en virgule flottante *Flair*, en Simple Précision, avec le signe +.

Mot 4 à 8 : oo oooo oooo + ou -

Cette carte doit être «NON CHARGEMENT»

Le système linéaire à résoudre sera perforé en Double Précision équation par équation (chaque équation comprend les premier et seconds membres, nous aurons, par conséquent, (*n* + p) coefficients par équation). Le modèle de carte utilisé sera le suivant :

Mot 1 NE NC 0 0 0 0 0 N +

avec NE : numéro de l'équation.

nous aurons : $1 \leqslant NE \leqslant n$

NC : numéro de carte de l'équation.

nous aurons : $1 \leqslant NC \leqslant \frac{n+p}{7} + (0 \text{ ou } 1)$

N : le nombre de coefficients en Double Précision, contenus dans la carte, multiplié par deux. Nous aurons : $1 \leqslant N \leqslant 6$

Mots 2, 4, 6 : partie gauche des nombres en virgule flottante Double Précision.

Mots 3, 5, 7 : partie droite des nombres en virgule flottante Double Précision.

Mot 8 : inutilisé

Toutes les cartes de ce modèle seront elles aussi, «Non chargement».

Remarque : Si l'on dispose des données en Simple Précision, il suffira de remplir tous les mots réservés aux parties droites des nombres en Double Précision avec :

00 0000 0000 + ou -

Forme du programme : Il est en «6 Mots» par carte (cartes roses). Le mot 8 est utilisé pour le contrôle du chargement du programme.

Le programme lui-même est précédé d'un programme de chargement (cartes jaunes) et immédiatement suivi d'une carte «Départ Calcul Automatique».

Le programme débute en 1351.

Encombrement du programme : Le programme proprement dit, occupe les mémoires 1300 à 1699. Les mémoires 1700 à 1999 sont occupées par l'A.D.P. (Arithmétique en virgule flottante Double Précision).

Le paquet decartes fourni constitue à la fois le programme proprement dit et l'A.D.F.

Déroulement du programme : Les interrupteurs du pupitre seront positionnés de la façon suivante :

STOP sur «effectif»
PROGRAMME sur «normal»
CONTROLE sur «marche»
INDICATEURS sur «registre programme»
DEPASSEMENT DE CAPACITE et ERREURS sur «arrêt».

Le tableau de connexions del'unité de lecture sera le tableau standard «8 mots».

On procèdera de la façon suivante :

Remplir le magasin de perforation et faire «Départ Perforation». Placer le programme dans le magasin de lecture, et immédiatement derrière la *Carte Caractéristique* et les cartes contenant les coefficients du système à résoudre. Afficher au pupitre :

70 1951 9999

Faire «Restauration Générale», «Départ Programme» et «Départ Lecture».

Une fois le programme enregistré, la lecture de la *Carte Caractéristique* et des données, ainsi que le départ des calculs se font sans intervention de l'opérateur.

Les calculs terminés, les résultats seront perforés et nous aurons au pupitre :

01 0999 1351

La machine est prête à traiter le système linéaire suivant, sans réenregistrement du programme.

Cartes «RESULTATS» : Chaque vecteur solution est perforé suivant le modèle de carte ci-dessous :

Mot 1 NS NC 0 0 0 0 0 N +

avec NS : numéro d'ordre du vecteur solution, nous aurons $1 \leq NS \leq p$

NC : numéro de carte pour le vecteur solution, nous aurons $1 \leq NC \leq \frac{n}{3} + (0 \text{ ou } 1)$

N : le nombre de racines en Double Précision, contenues dans la carte, multiplié par deux; nous aurons $1 \leq N \leq 6$

Mots 2, 4, 6 : partie gauche des racines en virgule flottante Double Précision.
Mots 3, 5, 7 : partie droite des racines en virgule flottante Double Précision.
Mot 8 : inutilisé.

Méthode utilisée : Il s'agit de la méthode «progressive» de JORDAN. C'est-à-dire que lors de la lecture de la k^o équation les $(k-1)$ équations précédentes ont subi $(k-1)$ éliminations. Le

système présente alors la forme suivante :

	(k-1)						(n+p) - (k-1)						
	1	0	0	0	0	0	x	x	x	x	x	x	
	0	1	0	0	0	0	x	x	x	x	x	x	
	0	0	1	0	0	0	x	x	x	x	x	x	
(k-1)	0	0	0	1	0	0	x	x	x	x	x	x	
	.....												
	.....												
	0	0	0	0	1	0	x	x	x	x	x	x	
	0	0	0	0	0	1	x	x	x	x	x	x	

Pour toutes les éliminations, le pivot est pris obligatoirement sur la diagonale principale.

Si l'on rencontre un pivot inférieur au minimum admis, l'équation en cause est perforée, et devra être replacée derrière le n° équation, dans le magasin de lecture.

Si tous les pivots possibles sont inférieurs au minimum admis, nous aurons l'ARRET :

01 9999 9999

Arrêts programmés :

01	0999	1351	Fin des calculs.
01	9999	9999	Pivots tous inférieurs au minimum admis.
01	0001	1611	La carte, que vient de lire la machine, n'est pas à sa place. Le contenu du Distributeur nous indique alors les références de la carte que la machine désire lire, le contenu de 8001 est :

00 0000 (NE) (NC)

Avec NE : numéro d'équation

NC : numéro de carte.

Il suffit alors de vider l'unité de lecture, et de remettre les cartes dans l'ordre convenable. En appuyant sur le bouton : « Départ Programme » la lecture reprendra normalement en 1611.

I, 5-2 NOTE EXPLICATIVE du PROGRAMME en SIMPLE PRECISION

MEMOIRES de TRAVAIL

1°) à contenus constants.

Nous désignerons par "n" l'ordre du système linéaire à résoudre et par "p" le nombre de séries de seconds membres proposées.

(1488) = oo (n + p) oooo

(1490) = 21 (n + p + 1) 1603

Ce contenu sera utilisé:

a) pour le calcul de l'instruction de stockage des résultats des (k - 1) éliminations effectuées sur la k° équation.

b) directement; pour la mise en mémoire des résultats de la k° élimination portant sur les (k - 1) premières équations.

(1509) = valeur absolue minimum admise pour le pivot.

(1524) = oo (n) oooo

(1625) = oo (n + p + 1) oooo

(1684) = 61 (n + p + 1) 1450

Ce contenu sera utilisé pour l'appel des coefficients des équations pivots pour les (k - 1) éliminations portant sur la k° équation.

(1751) = oo (p) oooo

2°) à contenus variables.

(1453) = multiplicateur, pour toutes les éliminations.

(1458) = oo (n + p - k + 1) oooo

le nombre "k" représente le nombre d'équations ayant fourni un pivot ou le nombre d'inconnues éliminées.

Nous avons: $1 \leqslant k \leqslant n$.

(1459) = 21 xxxx 1603

instruction de stockage des résultats des (k - 1) éliminations effectuées sur la k° équation.

Au début des calculs, nous avons toujours:

xxxx = (n+p-k+1)(k-1) + (n+p+1)

(1460) = perte maximum de précision.

Au début de la résolution de chaque système linéaire, cette mémoire est remise à zéro.

(1464) = 69 xxxx 1554

Ce contenu sera utilisé pour l'appel des coefficients:

- soit de la k° équation au cours de l'élimination de ses (k - 1) premières inconnues.

  Nous avons initialement, xxxx égal à:

  - k, pour l'élimination de la première inconnue.
  - (n+p-k+1)(k-1) + (n+p+1) dans les autres cas.

- soit des (k - 1) premières équations lors de l'élimination de leur k° inconnue.

(1466) = oo (k) oooo

(1473) = 67 xxxx 1605

lors de la mise en mémoire du pivot et de sa comparaison avec la valeur absolue minimum admise.

Nous avons: xxxx = adresse de la mémoire contenant le pivot.

= 60 xxxx 1500

avec xxxx = (n+p-k+1)(k-1) + (n+p+1) ; c'est à dire l'adresse de la mémoire contenant le pivot au début de la division de la k° équation par le pivot.

= 24 xxxx 1640

lors de la perforation: soit d'une équation ayant fourni un pivot inférieur, en valeur absolue, au minimum admis; soit des résultats.

Nous avons: xxxx = 1978 + le nombre de quantités à perforer.

(1481) = oo (n + p) oooo

lors du traitement de la première équation ou l'élimination de la première inconnue.

= oo oooo oooo

dans tous les autres cas.

(1482) = 60 xxxx 1500

Le contenu de cette mémoire sera utilisé pour l'appel des coefficients de la k° équation qui seront divisés par le pivot.

Nous avons: xxxx = (n+p-k+1)(k-1) + (n+p+2).

(1491) = 24 xxxx 1757

Cette instruction servira au chargement des (n + p) termes de chacune des équations.

Initialement, nous avons:

- xxxx = (n + p + 1) dans le cas de la première équation.
- xxxx = 0001, dans tous les autres cas.

(1503) = valeur exacte du pivot relatif à l'élimination de la k° inconnue,

(1510) = oo xxxx oooo

il s'agit du comptage des inconnues éliminées dans la k° équation.

Nous avons: $1 \leqslant xxxx \leqslant k$

(1514) = 21 xxxx 1704

Cette instruction permettra la mise en mémoire des quotients obtenus en divisant les termes de la k° équation par le pivot.

Initialement, nous avons:

xxxx = (n+p-k+1)(k-1) + (n+p+1)

(1532) = 69 xxxx 8002

(1539) = 69 yyyy 8002

Ces deux instructions serviront, à la fin de la k° élimination, au blocage de l'équation pivot à la suite des (k - 1) nouvelles équations.

Initialement, nous aurons:

xxxx = (n+p-k+1)(k-1) + (n+p+1)

yyyy = (n+p-k+1)(k-1) + 2n + 2p - k + 1

(1542) = oo xxxx oooo

avec "xxxx" représentant le numéro d'ordre du vecteur solution à perforer.

Nous aurons: $1 \leqslant xxxx \leqslant p$

(1547) = (1491) + oo (n + p) oooo

Ce contenu est utilisé pour tester la fin de l'enregistrement de chaque équation.

(1552) = instruction de SORTIE du Sous Programme de Perforation.

(1564) = 21 xxxx 1603

Cette instruction sera utilisée pour:

a) la mise en mémoire des résultats des (k-1) éliminations effectuées sur la k° équation.
Nous aurons, au début de chacune des (k - 1) éliminations:

xxxx = (n+p-k+1)(k-1) + (n+p+1)

b) le stockage des résultats de la k° élimination portant sur les (k - 1) premières équations.
Nous aurons initialement:

xxxx = n + p + 1

(1567) = (1600) + oo (n+p-k+1) oooo

Ce contenu permet de tester la fin de chacune des (k - 1) éliminations effectuées sur la k° équation.

(1574) = oo oooo (NE)(NC)

avec NE: numéro d'équation
NC: numéro de carte

Il s'agit de l'indicatif de la prochaine carte à lire pour le chargement d'une équation.

(1575) = oo xxxx oooo

En position facteur, nous avons le numéro d'équation, diminué de l'unité, à attribuer à la prochaine équation qui sera perforée.

(1600) = 61 xxxx 1450

Cette instruction sera utilisée pour appeler successivement tous les termes de l'équation pivot convenable:

a) lors de chacune des (k - 1) éliminations effectuées sur la k° équation.
Nous aurons au début de la première de ces éliminations:

xxxx = n + p + 1

b) lors de la k° élimination portant sur les (k - 1) premières équations.
Nous avons alors au début du traitement de chacune de ces équations:

xxxx = (n+p-k+1)(k-1) + (n+p+1)

(1607) = oo oooo (NE)oo

avec NE: numéro de la dernière équation enregistrée en mémoires.

(1609) = 61 xxxx 1450
avec xxxx = (n+p−k+1)(k−1) + (n+p+1)

(1634) = "pas" permettant de passer de l'adresse d'une mémoire à perforer à celle de la suivante.
Nous aurons:
(1634) = oo oool oooo
pour la perforation d'une équation; et
(1634) = oo (p) oooo
pour la perforation des vecteurs solutions.

(1649) = 24 xxxx 1757
Ce contenu servira au test de fin de carte, au cours de l'enregistrement d'une équation.
xxxx = l'adresse, augmentée de l'unité, de la mémoire où sera chargé le dernier mot de la carte.

(1659) = (1464) + oo (n+p−k) oooo
Ce contenu est utilisé pour tester la fin de la k° élimination pour chacune des (k − 1) premières équations.

(1669) = 60 xxxx 1500
xxxx = l'adresse, augmentée de l'unité, du dernier terme de la k° équation après l'élimination de ses (k − 1) premières inconnues.
Le contenu de la mémoire 1669 sera utilisé pour tester la fin de la division de la k° équation par le pivot.

(1699) = 69 xxxx 8003
instruction pour appeler les quantités à transferer dans les mémoires de perforation.

(1723) = 24 xxxx 1706
Cette instruction servira, à la fin de la k° élimination, au blocage de l'équation pivot à la suite des (k − 1) nouvelles équations.

(1751) = oo (k − 1) oooo

(1754) = oo xxxx oooo
"xxxx" représente successivement les adresses des mémoires contenant les premiers termes de chacune des (k − 1) premières équations.

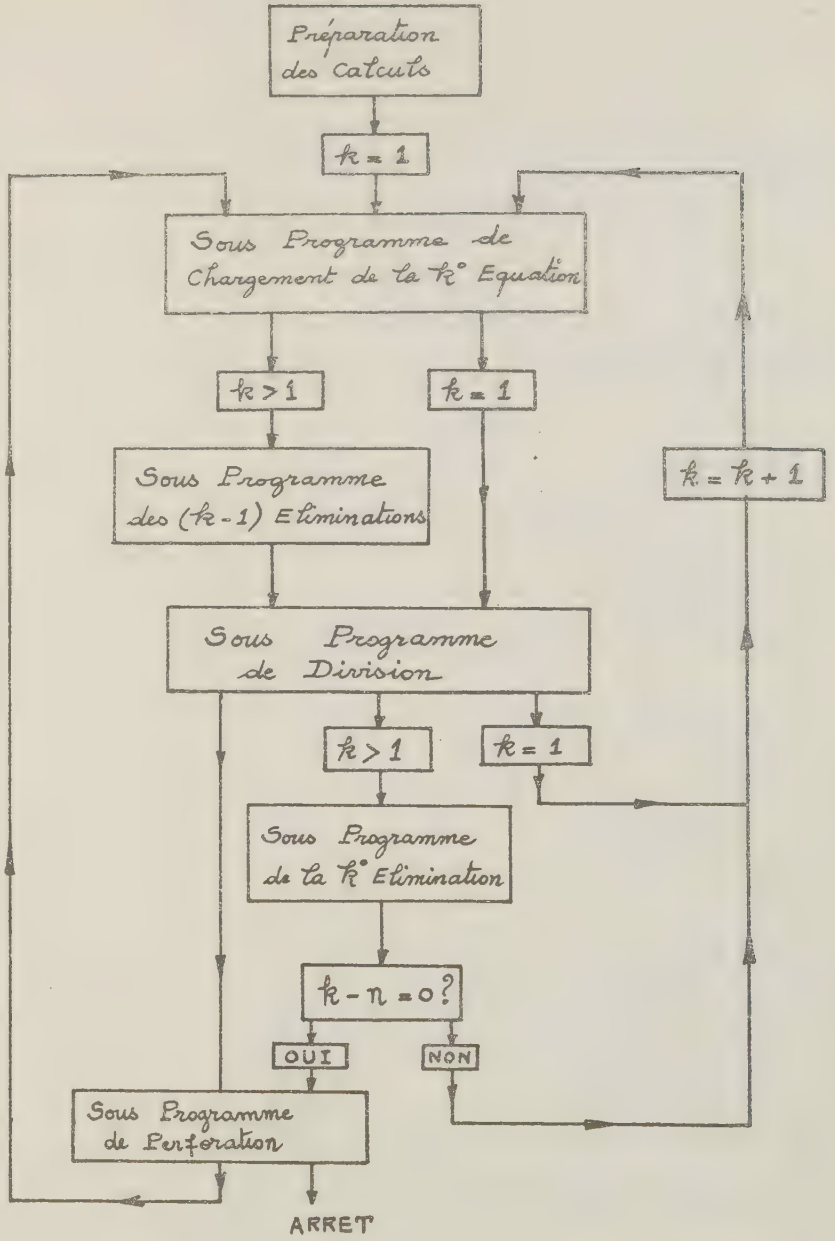
Au début de la k° élimination, nous avons:
: xxxx = n + p + 1
c'est à dire l'adresse du premier terme de la première équation.

(1769) = compteur des pivots inférieurs à la valeur absolue minimum admise.
Cette mémoire est remise à zéro au début de chacune des éliminations, c'est à dire avant la lecture d'une nouvelle équation.

(1853) = perte de précision relative à la dernière opération effectuée en Arithmétique Virgule Flottante.

PROGRAMME DE LIAISON

OBJET: Ce programme réalise, à l'aide de Sous Programmes convenables, la chaîne de calculs représentée par l'organigramme ci-après.



Préparation des calculs.

```
1651   70 1951 1701   Lecture de la carte caractéristique.
1701   69 1953 1658         (1953) = valeur absolue
                            minimum admise pour le pivot.
1658   24 1509 1662
1662   65 1951 1708         (1951) = oo ( n ) oooo
1708   24 1524 1680
1680   24 1575 1730
1730   21 1460 1616   Remise à zéro de la mémoire
                      contenant la perte maximum
                      de précision.
1616   24 1769 1622   Remise à zéro du compteur de pivots
                      inférieurs à la valeur absolue
                      minimum admise.
1622   24 1607 1610
1610   15 1952 1758         (1952) = oo ( p ) oooo
1758   24 1761 1666
1666   20 1488 1541         (1488) = oo (n+p) oooo
1541   24 1481 1584
1584   24 1458 1712
1712   15 1472 1531         (1472) = oo oool oooo
1531   24 1634 1537
1537   24 1466 1770         (1466) = oo (k=1) oooo
1770   20 1625 1581         (1625) = oo (n+p+1) oooo
1581   69 1684 1587         (1684) = 61 oooo 1450
1587   22 1684 1637         (1684) = 61 (n+p+1) 1450
1637   69 1490 1693         (1490) = 21 oooo 1603
1693   22 1490 1743         (1490) = 21 (n+p+1) 1603
1743   69 1459 1762         (1459) = 21 oooo 1603
1762   22 1459 1716         (1459) = 21 (n+p+1) 1603
1716   69 1525 1631         (1525) = 65 1459 1663
1631   24 1608 1766   Préparation de la SORTIE du
                      Sous Programme de Chargement.
1766   69 1672 1675         (1672) = 65 1725 1681
1675   24 1478 1496   Préparation de la SORTIE du
                      Sous Programme de Division.
```

1496
Sous Programme de CHARGEMENT
1608

A ce moment, nous avons:

(1608) = (1525)

il s'agit de la première instruction du Sous Programme de Division.

1525
Sous Programme de DIVISION
1478

```
(7) 1478   65 1725 1681         (1725) = 69 1700 1660
    1681   24 1608 1722   Modification de la SORTIE du
                          Sous Programme de Chargement.
                          Cette SORTIE restera inchangée
                          jusqu'à la fin des calculs.
    1722   21 1481 1734
    1734   69 1687 1540         (1687) = 69 1625 1632
    1540   24 1478 1498   Modification de la SORTIE du
                          Sous Programme de Division.
                          Cette SORTIE sera conservée
                          jusqu'à la fin des calculs.
(2) 1498   60 1466 1772   Première instruction effectuée
                          lors de la SORTIE du SOUS Programme
                          de la k° Elimination.
                                (1466) = oo ( k ) oooo
    1772   20 1769 1773   Remise à zéro du compteur de pivots
                          inférieurs à la valeur absolue
                          minimum admise.
    1773   11 1524 1731         (1524) = oo ( n ) oooo
    1731   44 1686 1736   Test de fin de tous les calculs.
```

Tous les calculs ne sont pas terminés.

```
    1686   10 8001 1545
```

```
1545    21 1751 1709        (1751) = oo (k-1) oooo
1709    10 1472 1582        (1472) = oo oool oooo
1582    24 1510 1624
1624    21 1466 1674        (1466) = oo ( k ) oooo
1674    60 1488 1793        (1488) = oo (n+p) oooo
1793    11 1751 1759        (1751) = oo (k-1) oooo
1759    21 1458 1496
```

1496
Sous-Programme de CHARGEMENT
1608

```
1608    69 1700 1660        (1700) = 11 1567 1521
(8) 1660    24 1467 1724    Préparation de la SORTIE du Sous
                            Programme d'Elimination.
```

Préparation de l'ENTREE dans le Sous Programme des (k-1) EliminationS.

```
1724    69 1684 1737        (1684) = 61 (n+p+1) 1450
1737    24 1600 1710
1710    65 1550 1760        (1550) = 69 oool 1554
1760    15 1751 1774        (1751) = oo (k-1) oooo
1774    20 1464 1775
1775    60 1458 1676        (1458) = oo (n+p-k+1) oooo
1676    19 1751 1726
1726    30 0004 1787        (8002) = oo [(n+p-k+1)(k-1)] oooo
1787    15 1490 1595        (1490) = 21 (n+p+1) 1603
1595    20 1459 1650
```

1550
Sous Programme des (k-1) ELIMINATIONS
1525

1525
Sous Programme de DIVISION
1478

Préparation de l'ENTREE dans le Sous-Programme de la k° Elimination.

```
1478    69 1625 1632        (1625) = oo (n+p+1) oooo
1632    24 1754 1776
1776    69 1451 1682        (1451) = 60 1464 1470
1682    24 1467 1732        Préparation de la SORTIE du
                            Sous-Programme d'Elimination.
1732    69 1490 1594        (1490) = 21 (n+p+1) 1603
1594    24 1564 1583
1583    65 1459 1633        (1459) = 21 [(n+p+1-k)(k-1) +n+p+1] 1603
1633    69 1609 1683        (1609) = 61 xxxx 1450
1683    22 1609 1750
```

1750
Sous-Programme de la k° ELIMINATION
1498

```
1498                (2)
```

Tous les calculs sont terminés. Nous allons perforer les résultats.
Il s'agit de perforer colonne par colonne, une matrice pxn qui réprésente les 'p' vecteurs solutions correspondant aux 'p' vecteurs seconds membres.
Ainsi pour perforer le i° vecteur solution, nous savons que:

- le nombre de termes à perforer est égal à 'n'.
- le premier terme se trouve dans la mémoire d'adresse (n+p+1+i).
- l'on passe d'un terme au suivant en ajoutant p à son adresse.

```
1736    65 1761 1733        (1761) = oo ( p ) oooo
1733    24 1634 1538        (8003) = oo oooo oooo
(1) 1538    10 1472 1588    (1472) = oo oool oooo
1588    21 1542 1645        (1542) = numéro d'ordre du
                            vecteur solution à perforer.
```

Préparation de l'ENTREE dans le Sous-Programme de Perforation.

```
1645    35 0004 1638
1638    21 1977 1688
1688    65 1488 1644        (1488) = oo (n+p) oooo
1644    10 1524 1738        (1524) = oo ( n ) oooo
1738    15 1542 1597        (1542)
1597    69 1502 1549        (1502) = 60 1542 1647
```

1549
Sous-Programme de PERFORATION
1552

```
1552    60 1542 1647        (1542) = numéro d'ordre du
                            vecteur solution à perforer.
1647    11 1761 1788        (1761) = oo ( p ) oooo
1788    44 1591 1592    Test de fin de perforation des
                        résultats.
```

Tous les vecteurs solutions n'ont pas été perforés.

```
1591    10 8001 1538    (1)
```

La perforation des résultats est terminée.

```
1592    71 1777 1639        Perforation d'une carte donnant
                            - en mot 1: la perte maximum
                            de précision en position
                            adresse instruction suivante,
                            et l'élimination au cours de
                            laquelle elle a eu lieu.
                            - dans les mots 2 à 8:
                                  oo oooo oooo -
1639    01 0999 1651        ARRET - La résolution du
                            système linéaire proposé est
                            terminée. Le programme est
                            prêt pour une nouvelle
                            résolution.
```

E T U D E

D E S

S O U S - P R O G R A M M E S

SOUS PROGRAMME DE CHARGEMENT

OBJET: Chargement d'une équation (n+p termes) à partir de la mémoire d'adresse (n+p+1) pour la première équation, d'adresse 0001 pour toutes les autres équations.
ENTREE: Indifférente.

Préparation des tests de fin de chargement et de fin de carte.

```
      1496   65 1599 1756      (8002) = 24 0001 1757
      1756   15 1481 1636      (1481) = 00 (n+p) 0000
                                lors du chargement de la
                                première équation.
                                     = 00 0000 0000
                                lors du chargement des
                                autres équations
      1636   20 1491 1494       1491
      1494   15 1488 1643      (1488) = 00 (n+p) 0000
      1643   20 1547 1501       1547
```

Détermination de l'indicatif de la carte à lire.

```
      1501   60 1607 1611      (1607) = 00 0000 (NE)00
                                NE: numéro de l'équation
                                    précédente.
      1611   10 1620 1598      (1620) = 00 0000 0100
      1598   21 1607 1560      (1607
(3)   1560   10 1765 1670      (1765) = 00 0000 0001
      1670   21 1574 1530       1574
```

Lecture et chargement.

```
(4)   1530   70 1951 1551
      1551   65 1951 1657
      1657   35 0004 1720
      1720   11 1574 1580      (1574) = indicatif de la carte
                                à lire, en position I
      1580   44 1533 1534
```

Nous n'avons pas lu la carte désirée.

```
      1533   01 0001 1530  (4)
```

La lecture est correcte.

```
      1534   15 1491 1495
      1495   20 1649 1452       1649
      1452   65 1707 1661      (1707) = 69 1952 8003
      1661   10 1491 8002
(5)   8002   69 1952 8003
      8003   24 xxxx 1757
      1757   10 1472 1630      (1472) = 00 0001 0000
      1630   15 8001 1589
      1589   11 1547 1601
      1601   44 1558 1608  Test de fin de chargement.
```

Le chargement n'est pas terminé.

```
      1558   10 8001 1771
      1771   11 1649 1508
      1508   44 1711 1612  Test de fin de carte.
```

La carte n'est pas terminée.

```
      1711   10 8001 8002  (5)
```

La carte est terminée.

```
      1612   24 1491 1544
      1544   60 1574 1560  (3) modification de l'indicatif
                                 pour la lecture de la carte
                                 suivante.
```

Le chargement est terminé.

```
      1608   SORTIE : sous la dépendance du programme de
                      liaison.
```

SOUS PROGRAMME DE DIVISION

OBJET: Ce sous programme divise l'équation pivot par le pivot.
L'adresse du pivot se trouve en position facteur du contenu de la mémoire 1459 .
Tous les termes de l'équation se trouvent en séquence à la suite du pivot.
Au cours de la k° élimination, le nombre de termes de l'équation pivot est égal à (n+p-k+1)
La mise en mémoire des résultats se fait à partir de celle occupée par le pivot.

ENTREE: Indifférente.

Mise en mémoire du pivot et comparaison avec la valeur minimum admise.

```
1525   65 1459 1663
1663   69 1516 1619      (1516) = 67 oooo 1605
1619   22 1473 8001
8001   67 xxxx 1605
1605   24 1503 1606
1606   16 1509 1713      (1509) = valeur minimum admise
                                  pour le pivot en valeur absolue.
1713   46 1566 1717
```

La valeur absolue du pivot est supérieure au minimum admis.
Préparation des divisions et du test de fin de calcul.

```
1717   65 1459 1763
1763   69 1514 1767      (1514) = 21 oooo 1704
1767   22 1514 1518
1518   69 1621 1474      (1621) = 60 oooo 1500
1474   22 1473 1526
1526   60 8001 1535
1535   15 8001 1493
1493   15 1458 1614      (1458) = oo (n+p-k+1) oooo
1614   20 1669 1522
```

(6) 1522 10 1472 1677 (1472) = oo oool oooo
1677 21 1482 1585
1585 11 1669 1523
1523 44 1482 1478

Les calculs ne sont pas terminés, nous allons effectuer la division suivante.

1482 60 xxxx 1500
Séquence d'éntrée en A.V.F. pour une division.

```
1500   69 1503 1506
1506   24 1801 1504
1504   69 1900 1553      VDV
1553   24 1808 1511
1511   69 1514 1800
```

```
1800
   Sous-Programme
     A. V. F.
          1803
```

(1803) = (1514)
= 21 xxxx 1704

Mise en mémoire du résultat.

1803 21 xxxx 1704

Préparation de la division suivante.

```
1704   65 1514 1719
1719   15 1472 1727      (1472) = oo oool oooo
1727   20 1514 1568
1568   60 1482 1522  (6)
```

La valeur absolue du pivot est inférieure au minimum admis.

```
1566   60 1769 1573      (1769) = compteur des pivots
                                  examinés au cours de la k° élimination
1573   10 1466 1671      (1466) = oo (k  ) oooo
1671   11 1524 1479      (1524) = oo ( n ) oooo
1479   44 1483 1484
```

Tous les pivots possibles n'ont pas été examinés.

```
1483   60 1769 1623
1623   10 1472 1528
1528   21 1769 1572
```

Préparation de la perforation de l'équation n'ayant pas fourni de pivot.
Le modèle de carte obtenu sera identique à celui des cartes "données". Un nouveau numéro sera attribué à l'équation ainsi perforée , il s'agit du contenu de la mémoire 1575

```
1572    60 1575 1529
1529    10 1472 1578        (1472) = oo oool oooo
1578    24 1510 1664
1664    21 1575 1628
1628    35 0004 1489
1489    21 1977 1480
1480    65 1472 1678
1678    15 1481 1635        (1481) = oo (n+p) oooo
                            pour la première élimination.
                                   = oo oooo oooo
                            pour les autres éliminations.
1635    10 1488 1543        (1488) = oo (n+p) oooo
1543    69 1496 1549        (1496) = première instruction
                            du sous programme de chargement.
```

1549
Sous Programme de PERFORATION
1552

A ce moment, nous avons:
(1552) = (1496)
il s'agit de la première instruction du Sous Programme de Chargement.

1496
Sous Programme de CHARGEMENT
1608

Il y a lieu de distinguer deux cas.

1°) Cas de la première élimination.

Nous avons alors:

(1608) = (1525) ; il s'agit de la première instruction du Sous Programme de Division.

1525
Sous Programme de DIVISION
1478

1478 Retour au Programme de Liaison
Voir en (7), page 4

2°) Cas des autres éliminations

Nous avons alors:

(1608) = (1725)

```
1608    69 1700 1660        (1700) = 11 1567 1521
1660    Retour au Programme de Liaison.
        Voir en (8), page 5
```

Tous les pivots possibles pour la k° élimination ont été examinés.

1484 01 9999 9999

Toutes les divisions sont terminées.

1478 SORTIE , sous la dépendance du programme de liaison.

SOUS PROGRAMME d'ELIMINATION

OBJET Soient trois nombres A, B et C en virgule flottante FLAIR. Les nombres A et C sont dans des mémoires dont les adresses variables sont respectivement indiquées par les facteurs des contenus des mémoires 1600 et 1464. Le nombre B, qui peut être le même pour toute une série de calculs, se trouve dans la mémoire 1453.

Ce Sous Programme effectue, à l'aide de l'A.V.F. , l'opération $-AxB+C$, met le résultat dans une mémoire d'adresse variable indiquée par le facteur du contenu de la mémoire 1564 et finalement prépare l'opération suivante.

De plus, ce Sous Programme détecte la perte maximum de précision rencontrée au cours des calculs.

ENTREE Indifférente.

SORTIE Sous la dépendance du Programme de Liaison.

```
1600    61 xxxx 1450
```

Séquence d'Entrée en A.V.F. pour effectuer une multiplication.

```
1450    69 1453 1456
1456    24 1801 1454
1454    69 1996 1499        VMU
1499    24 1808 1461
1461    69 1464 1800
```

```
1800
   Sous-Programme
      A. V. F.
                1803
```

(1803) = (1464)
= 69 xxxx 1554

Séquence d'Entrée en A.V.F. pour effectuer une addition.

```
1803    69 xxxx 1554
1554    24 1801 1604
1604    69 1902 1455        VAA
1455    24 1808 1561
1561    69 1564 1800
```

```
1800
   Sous-Programme
      A. V. F.
                1803
```

(1803) = (1564)
= 21 xxxx 1603

```
1803    21 xxxx 1603
```

Etude de la perte de précision.

```
1603    60 1460 1465
```

(1460) = perte de précision maximum rencontrée depuis le début des calculs.

```
1465    11 1853 1457
```

(1853) = perte de précision de la dernière opération.

```
1457    46 1477 1468
```

La perte de précision de la dernière opération est supérieure aux pertes antérieures.

```
1477    24 1460 1469
1469    60 8001 1463
1463    10 1466 1471
```

(1466) = compteur des éliminations.

```
1471    21 1777 1468
```

Nous venons d'enregistrer la dernière perte de précision.
Ou bien, la perte de précision de la dernière opération est inférieure aux pertes antérieures.
Dans les deux cas, préparation de l'opération suivante.

```
1468    60 1600 1505
```

```
1505    15 1564 1519
1519    10 1472 1527        (1472) = oo oool oooo
1527    15 8001 1485
1485    21 1600 1653
1653    20 1564 1467
1467    SORTIE, sous la dépendance du Programme de Liaison.
```

SOUS PROGRAMME DES (k-1) ELIMINATIONS

OBJET: Quand nous lisons la k° équation, les (k-1) premières inconnues des (k-1) premières équations ont déjà été éliminées.

Ce sous programme a pour but d'éliminer les (k-1) premières inconnues de la k° équation. Les (k-1) premiers coefficients de cette équation seront les multiplicateurs à utiliser lors des (k-1) éliminations. Les calculs d'élimination ne porterons que sur les (n+p-k+1) derniers coefficients de cette équation.

Les (k-1) équations pivots, composées chacune de (n+p-k+1) termes, se trouvent en séquence à partir de la mémoire d'adresse (n+p+1). La k° équation (k≠1) est chargée à partir de la mémoire d'adresse 0001.

ENTREE: Trois mémoires doivent être chargées avant d'entrer dans ce sous programme:

```
1600    avec    61 (n+p+1) 1450
1464    ""      69 (  k  ) 1554
1459    ""      21 [(n+p-k+1)(k-1)+n+p+1] 1603
```

Mise en mémoire du multiplicateur.

```
1650    65 1703 1507        (1703) = 69 oooo 1753
1507    15 1510 8002        (1510) = initialement
                                     oo oool oooo
8002    69 xxxx 1753
1753    24 1453 1556
```

Préparation de la mise en mémoires des résultats et du test de fin d'une élimination.

```
1556    69 1459 1462
1462    24 1564 1517
1517    65 1600 1555
1555    15 1458 1513        (1458) = oo (n+p-k+1) oooo
```

```
      1513    20 1567 1600
(9)   1600
        Sous-Programme
        d'ELIMINATION
                        1467
```

A ce moment, nous avons:
(1467) = (1700)

```
      1467    11 1567 1521
      1521    44 1475 1476    Test de fin d'une élimination.
```

L'élimination en cours n'est pas terminée.

```
      1475    65 1464 1569
      1569    15 1472 1577        (1472) = oo oool oooo
      1577    20 1464 1600  (9)
```

L'élimination en cours est terminée.
Préparation de l'éliminationsuivante.

```
      1476    24 1600 1654
      1654    65 1459 1563
      1563    69 1464 1617
      1617    22 1464 1667
      1667    65 1510 1515
      1515    15 1472 1627
      1627    20 1510 1613
      1613    16 1466 1571        (1466) = oo ( k ) oooo
      1571    45 1650 1525  Test de fin des (k-1) éliminations.
```

Les éliminations ne sont pas terminées.

```
      1650    Nous effectuons l'élimination suivante.


      1525    Entrée dans le sous programme de Division.
```

SOUS PROGRAMME DE LA k° ELIMINATION

OBJET: Les (k-1) premières inconnues de la k° équation ont été éliminées. Puis les (n+p-k+1) termes restant ont été divisés par le coefficient de la k° inconnue (le pivot). L'équation ainsi obtenue sera l'équation pivot de la k° élimination. Elle est constitué de (n+p-k) termes chargés à partir de la mémoire d'adresse (n+p-k+1)(k-1)+n+p+1 (le coefficient de la k° inconnue égal à l'unité n'est plus conservé en mémoire).

Ce sous programme a pour but d'effectuer la k° élimination, puis de regrouper les résultats de façon à préparer les calculs ultérieurs.

La k° élimination portera sur (k-1) équations rangées en séquence à partir de la mémoire d'adresse (n+p+1). Chaque équation est composée de (n+p-k+1) termes; le premier de ceux-ci est le coefficient de la k° inconnue que l'on élimine, et sera par conséquent "multiplicateur".

ENTREE: Trois mémoires doivent chargées de la façon suivante:

```
      1754    avec    00 (n+p+1) 0000
      1609     ""     61 [(n+p-k+1)(k-1)+n+p+1] 1450
      1564     ""     21 (n+p+1) 1603
```

Mise en mémoire du multiplicateur

```
(10)  1750    65 1754 1559
      1559    10 8001 1618
      1618    15 1721 8002        (1721) = 69 oooo 1655
      8002    69 0xxx 1655
      1655    24 1453 1656        (1453) = multiplicateur
```

Préparation du test de fin d'équation/

```
      1656    69 1609 1512
      1512    24 1600 1705
      1705    15 1458 1714        (1458) = oo (n+p-k+1) oooo
```

```
      1714    10 1550 1755        (1550) = 69 0001 1554
      1755    22 1659 1562
(11)  1562    21 1454 1600
```

1600
Sous-Programme d'ELIMINATION
1467

A ce moment, nous avons:
(1467) = (1451)

```
      1467    60 1464 1470
      1470    10 1472 1728        (1472) = oo oool oooo
      1728    11 1659 1764
      1764    44 1668 1718    Test de fin d'équation.
```

Tous les termes de l'équation n'ont pas été utilisés.

```
      1668    10 8001 1562    (11)
```

Tous les termes de l'équation ont été utilisés
Les (k-1) équations ont-elles été utilisées?

```
      1718    65 8001 1579
      1579    69 8003 1486
      1486    22 1754 1557
      1557    60 8001 1565
      1565    10 1768 1673        (1768) = 61 oooo 1450
      1673    11 1609 1615
      1615    44 1750 1520    Test de fin de la k° élimination.
```

Les (k-1) équations n'ont pas toutes été utilisées.
Nous passons à l'équation suivante.

```
      1750                    (10)
```

La k° élimination est terminée. Nous allons bloquer la k° équation (équation pivot) à la suite des (k-1) nouvelles équations que nous venons d'obtenir

```
      1520    65 8001 1629
      1629    69 1532 1685        (1532) = 69 xxxx 8002
      1685    22 1532 1735
      1735    15 1458 1665        (1458) = oo (n+p-k+1) oooo
      1665    16 1472 1679        (1472) = oo oool oooo
      1679    69 1532 1536
      1536    22 1539 1492
      1492    65 1564 1570        (1564) = en position facteur:
                                  l'adresse augmentée de un du dernier
                                  résultat enregistré.
      1570    69 1723 1576        (1723) = 24 xxxx 1706
      1576    22 1723 1626
      1626    65 8001 1586
      1586    10 1532 8001
(12)  8001    69 xxxx 8002
      8002    24 xxxx 1706
      1706    10 1472 1729        (1472) = oo oool oooo
      1729    15 8001 1487
      1487    11 1539 1593
      1593    44 1497 1498    Test de fin de blocage de la k°
                              équation.
```

Tous les termes de la k° équation n'ont pas été déplacés.

```
      1497    10 8001 8003    (12)
```

Le blocage de la k° équation est terminé.

```
      1498    SORTIE, nous retournons dans le programme de
              Liaison.
```

SOUS PROGRAMME DE PERFORATION

OBJET: Ce sous programme répond à deux buts:
- perforer une équation n'ayant pu fournir un pivot supérieur en valeur absolue au minimum admis.
- perforer les résultats.

ENTREE: Nous placerons en

8001 une instruction de sortie.

8002 en position facteur: l'adresse de la première mémoire à perforer.

8003 en position facteur: le nombre de mémoires à perforer.

Lors de la perforation, nous passerons d'une mémoire à la suivante en ajoutant à l'adresse de la première une quantité constante qui sera placée en position facteur de la mémoire 1634 .

L'indicatif caractérisant les mémoires perforées sera placé dans les positions 9 et 10 de la mémoire 1977.

SORTIE; L'instruction placée en 8001 lors de l'entrée dans le sous programme est exécutée.

```
      1549   24 1552 1689
      1689   69 1642 1695      (1642) = 69 oooo 8003
      1695   22 1699 1602
      1602   10 1739 1694      (1739) = 24 1978 1640
      1694   21 1473 1789
(13)  1789   65 1699 1590
      1590   10 1739 8002
(14)  8002   69 xxxx 8003
      8003   24 19xx 1640
      1640   10 1472 1690      (1472) = oo oool oooo
      1690   15 1634 1740      (1634) = pas des mémoires à
                                      perforer/
      1740   11 1473 1790
      1790   44 1744 1794  Test de fin de perforation.
```

La perforation n'est pas terminée.

```
      1744   10 8001 1641
      1641   11 1745 1749      (1745) = 24 1985 1640
      1749   44 1711 1691  Test de fin de carte.
```

Les mots 2 à 8 de la carte en cours de perforation n'ont pas été tous remplis.

```
      1711 10 8001 8002 | (14)
```

La carte est remplié, il faut la perforer.

```
      1691   20 1699 1652
      1652   60 1473 1741
      1741   11 1795 1799      (1795) = oo ooo7 oooo
      1799   21 1473 1791
```

Composition du contenu de la mémoire 1977: mot 1 de perforation.

```
      1791   60 1977 1692
      1692   10 1715 1742      (1715) = oo oloo oooo
      1742   15 1546 1702      (1546) = oo oooo ooo7
      1702   69 8003 1792
      1792   23 1977 1596
      1596   71 1977 1789 | (13)
```

La perforation sera terminée après la perforation de la dernière carte en cours de remplissage.

Compisition du contenu de la mémoire 1977.

```
      1794   60 8001 1646
      1646   11 1739 1696      (1739) = 24 1978 1640
      1696   65 8003 1746
      1746   35 0004 1796
      1796   10 1977 1697
      1697   10 1715 1747
      1747   69 8003 1797
      1797   23 1977 1548
      1548   71 1977 1552
      1552   SORTIE, éxécution de l'instruction placée en
             8001 lors de l'entrée dans ce sous programme.
```

I, 6 SOUS-PROGRAMMES de RESOLUTION de SYSTEMES LINEAIRES à MATRICE-BANDE en SIMPLE PRECISION, et en DOUBLE PRECISION . Méthode de JORDAN Progressive.

(Références : JSP 3 et JDP 3).

I, 6-1 NOTICES d'UTILISATION

I, 6-1-1 SOUS-PROGRAMME en SIMPLE PRECISION

Objet : Etant donné la Matrice-Bande et les vecteurs seconds membres d'un système linéaire, ce Sous-Programme calcule en Virgule Flottante *Flair* et en Simple Précision, les vecteurs solutions.

Conditions d'Utilisation : Si «*n*» désigne l'ordre de la Matrice-Bande du système linéaire, «m» le nombre maximum de termes de la bande, et «p» le nombre de vecteurs seconds membres, la limite d'utilisation de ce Sous-Programme est :

$$(m+p)\ n - \frac{m^2-1}{4} \quad < 1549$$

Ainsi avec $p=1$ et $m=3$, nous avons $n_{MAX} = 387$.

La Virgule Flottante simulée étant du type *Flair*, on représentera :

1 par 51 1000 0000 +

Il faudra enregistrer l'A V F (Arithmétique en Virgule Flottante) sur le tambour.

Utilisation : Le système à Matrice-Bande à résoudre doit être rangé, équation par équation, dans des mémoires consécutives du tambour. Chaque équation se composant des termes du premier membre situés dans la bande et des «p» seconds membres. Un équation comprendra au minimum $(\frac{m+1}{2} + p)$ termes et au maximum $(m+p)$ termes.

Le calcul n'utilise aucune mémoire de travail. Par suite, le premier membre de la limite d'utilisation énoncée précédemment, exprime le nombre de mémoires nécessaires pour l'enregistrement des termes non nuls d'un système linéaire à Matrice-Bande.

Si nous désignons par «M» l'adresse de la première mémoire du bloc où est rangé le système linéaire à Matrice-Bande (équation par équation), l'utilisateur devra placer en :

8003 M *n* 0 0

8002 m p 0 0 0 0

8001 une instruction de SORTIE.

Après les calculs, un bloc de p x *n* mémoires consécutives à partir de l'adresse «M», contient les «p» vecteurs solutions. Ces vecteurs solutions sont rangés de telle façon que le j^o vecteur solution correspondant au j^o vecteur second membre, occupe les mémoires d'adresses :

M + j - 1

M + j - 1 + p

M + j - 1 + 2 p

. . . . . .

M + j - 1 + (*n* - 1) p

Ainsi, par exemple, si nous avons :

$n = 7$, $m = 3$, $p = 2$, et M = 0123

Les mémoires 0123 à 0155 seront réservées aux données (Matrice-Bande et seconds membres).

La 3° ligne ou équation du système sera enregistrée dans les mémoires 0132 à 0136.

Les résultats occuperont pour :

le 1° vecteur solution, les mémoires : 0123, 0125, 0127, 0129, 0131, 0133, 0135 ;

le 2° vecteur solution, les mémoires : 0124, 0126, 0128, 0130, 0132, 0134, 0136.

Forme du sous-programme : L'utilisateur dispose de deux paquets de cartes dont les contenus devront être enregistrés sur le tambour.

1°/ *l'A V F* (Arithmétique en Virgule Flottante) : Ce programme occupe les mémoires 1800 à 1999 sauf les zones entrée et sortie standard de la bande 1950 (1950 à 1960 et 1977 à 1986).

Il se présente sous forme de cartes «chargement 7 mots par carte» (cartes roses) et est précédé d'un programme de chargement qui utilise la bande 0000. Il faudra donc prévoir son chargement en temps utile dans le cas où l'on utiliserait la bande 0000 pour stocker les données ou à toute autre fin.

2°/ *Le SOUS-PROGRAMME* de Résolution de Systèmes Linéaires à Matrice-Bande, en Simple Précision.

Il se présente sous forme numérique translatable «1 mot par carte» (cartes blanches). Le facteur de translation Δ qui peut être quelconque mais inférieur ou égal à 1550, sera chargé en 1950. La première instruction de ce Sous-Programme se trouve en Δ + 0000.

Remarques : Lors du déroulement des calculs, nous aurons les boutons :

STOP sur «effectif»

DEPASSEMENT de CAPACITE et ERREUR sur «arrêt».

Encombrement : 200 mémoires pour l'A V F (obligatoirement de 1800 à 1999, les zones d'entrée et de sortie standard de la bande 1950 étant disponibles).

250 mémoires pour le Sous-Programme ;

Méthode utilisée : Il s'agit de la méthode PROGRESSIVE de JORDAN. C'est-à-dire que, lors de la prise en considération de la k° équation, les (k - 1) premières inconnues ont déjà été éliminées dans les k (k - 1) premières équations.

Nous éliminerons d'abord les (k - 1) premières inconnues dans la k° équation à l'aide des (k - 1) premières équations jouant le rôle d'équations pivots. Ensuite, la k° équation ainsi transformée, servira d'équation pivot pour l'élimination de la k° inconnue dans les (k - 1) premières équations.

Nous opérerons de cette façon, jusqu'à l'utilisation de la n° équation, et l'élimination de la n° inconnue.

Pour toutes les éliminations, le pivot est pris obligatoirement sur la diagonale principale.

Si l'on rencontre un pivot nul, nous avons l'arrêt :

01 9999 9999

Il est alors impossible de poursuivre les calculs.

Remarques : Ce Sous-Programme détecte la perte maximale de précision rencontrée en cours de chaque résolution de système linéaire à Matrice-Bande.

A la fin des calculs, nous trouverons cette perte maximale de précision dans la mémoire + 0030. Le contenu de cette mémoire sera :

00 (k) 000N

Ce qui signifie qu'au cours de l'élimination de la k° inconnue nous avons eu une perte de précision de N chiffres.

Si le contenu de $\Delta + 0030$ est nul, aucune perte de précision n' a été détectée.

Arrêts programmes :

01	9999	9999	Pivot nul.
01	0111	8000	Arrêt OVERFLOW. L'exposant du résultat d'une opération en arithmétique flottante est supérieur à 99.
01	0222	8000	Lors d'une division, nous avons un diviseur nul.

I, 6-1-2 SOUS-PROGRAMME en DOUBLE PRECISION

Objet : Etant donné la Matrice-Bande et les vecteurs seconds membres d'un système linéaire, ce Sous-Programme calcule en Virgule Flottante *Flair* et en DOUBLE PRECISION, les vecteurs solutions.

Conditions d'Utilisation : Si « *n* » désigne l'ordre de la Matrice-Bande du système linéaire : « m » le nombre maximal de termes de la bande, et « p » le nombre de vecteurs seconds membres, la limite d'utilisation de ce Sous-Programme est :

$$2\,(m+p)\,n - \frac{m^2-1}{2} \leqslant 1399$$

Ainsi avec $p = 1$ et $m = 3$, nous avons : $n_{MAX} = 175$

La Virgule Flottante simulée étant du type *Flair* , on représentera :

1 par $\underbrace{51\quad 1000\quad 0000}_{I}\quad \underbrace{00\quad 0000\quad 0000}_{II}$

Dans tout ce qui suit, nous appelleront I, la partie gauche et II, la partie droite d'un nombre en Virgule Flottante Double Précision.

Il faudra enregistrer l' A.D.P. (Arithmétique en Virgule Flottante Double Précision), sur le tambour.

Utilisation : Le système à Matrice-Bande à résoudre doit être rangé , équation par équation, dans des mémoires consécutives du tambour. Chaque équation se composant des termes du premier membre appartenant à la bande et des « p » seconds membres. Une équation comprendra au minimum $(\frac{m+1}{2} + p)$ termes et au maximum $(m + p)$ termes.

Le rangement du système à Matrice-Bande en DOUBLE PRECISION, se fera de la façon suivante :

- Un premier bloc de $[(m+p)\,n - \frac{m^2-1}{4}]$ mémoires dont la première a pour adresse M, contiendra toutes les parties gauches des nombres en DOUBLE PRECISION.
- Un second bloc de $[(m+p)\,n - \frac{m^2-1}{4}]$ mémoires dont la première a pour adresse : M + à (S O) contiendra toutes les parties droites des nombres en Double Précision.

Remarque : Nous noterons qu'il faut $S \geqslant [(m+p)\,n - \frac{m^2-1}{4}]$

D'autre part, dans le cas où les données sont en Simple Précision, il ne faut pas oublier de s'assurer que les

mémoires devant contenir, au cours des calculs, les parties droites des nombres en Double Précision sont bien initialement soit à zéro +, soit à zéro -.

Avec les notations déjà définies, pour entrer dans le Sous-Programme de Résolution des Systèmes Linéaires à Matrice-Bande en DOUBLE PRECISION, l'utilisateur devra placer en :

8003 M S

8002 n m p

8001 une instruction de SORTIE

Après les calculs :

- un premier bloc de p.n mémoires consécutives à partir de l'adresse M, contient les parties gauches des «p» vecteurs solutions en Double Précision.
- un second bloc de p.n mémoires consécutives à partir de l'adresse (M + S), contient les parties droites des «p» vecteurs solutions en Double Précision.

Les vecteurs solutions sont rangés de telle façon, que le j° vecteur solution correspondant au j° vecteur second membre, occupe les mémoires d'adresses :

- pour les parties gauches des résultats en Double Précision :

M + j - 1

M + j - 1 + p

M + j - 1 + 2 p

. . . . . .

M + j - 1 + (n - 1) p

- pour les parties droites des résultats en Double Précision :

M + S + j - 1

M + S + j - 1 + p

M + S + j - 1 + 2 p

. . . . . .

M + S + j - 1 + (n - 1) p

Ainsi, par exemple, si nous avons :

n = 7, m = 3, p = 2 et M = 0123, S = 0111

Les mémoires 0123 à 0155 (parties gauches) et 0234 à 0266 (parties droites) seront réservées aux données en Double Précision (Matrice-Bande et seconds membres).

Les résultats occuperont :

- pour le 1° vecteur solution, les mémoires : 0123, 0125, 0127, 0129, 0131, 0133, 0135, 0234, 0236, 0238, 0240, 0242, 0244, 0246 ;
- pour le 2° vecteur solution, les mémoires : 0124, 0126, 0128, 0130, 0132, 0134, 0136, 0235, 0237, 0239, 0241, 0243, 0245, 0247 ;

Formes du sous-programme : l'utilisateur dispose de deux paquets de cartes dont les contenus devront être enregistrés sur le tambour.

1° / *l'A.D.F.* (Arithmétique en virgule flottante Double Précision). Ce programme occupe les mémoires 1700 à 1999 sauf les zones entrée et sortie standard de la bande 1950 (1950 à 1960 et 1977 à 1986).
Il se présente sous forme de cartes «chargement 7 mots par carte» (cartes roses) et est précédé d'un programme de chargement (cartes jaunes) qui utilise la bande 0000. Il faudra donc prévoir son chargement en temps utile dans le cas où l'on utilisera la bande 0000 pour stocker les données ou à toute autre fin.
2° / *Le Sous-Programme* de Résolution des Systèmes Linéaires à Matrice-Bande, en Double Précision. Il se présente sous forme numérique translatable «1 Mot par carte» (cartes blanches). Le facteur de translation Δ qui peut être quelconque mais inférieur ou égal à 1400, sera chargé en 1950.

La première instruction de ce Sous-Programme se trouve en Δ + 0000.

Remarque : Lors du déroulement des calculs, nous aurons les boutons :

STOP sur «effectif»

DEPASSEMENT de CAPACITE et ERREURS sur «arrêt»

Encombrement : 300 mémoires pour l'A.D.F. (obligatoirement de 1700 à 1999, les zones d'entrée et de sortie standard de la bande 1950 étant disponibles).

300 mémoires pour le Sous-Programme.

Méthode utilisée : Il s'agit de la méthode PROGRESSIVE de JORDAN. C'est-à-dire que lors de la prise en considération de la k° équation, les (k - 1) premières inconnues ont déjà été éliminées dans les (k - 1) premières équations.

Nous éliminerons d'abord les (k - 1) premières inconnues dans la k° équation à l'aide des (k - 1) premières équations jouant le rôle d'équations pivots. Ensuite, la k° équation ainsi transformée, servira d'équation pivot pour l'élimination de la k° inconnue dans les (k - 1) premières équations.

Nous opérerons de cette façon, jusqu'à l'utilisation de la n° équation et l'élimination de la n° inconnue.

Pour toutes les éliminations, le pivot est pris obligatoirement sur la diagonale principale.

Si l'on rencontre un pivot nul, nous avons l'arrêt :

01 9999 9999

et il est alors impossible de poursuivre les calculs.

I, 6-2 NOTE EXPLICATIVE du SOUS-PROGRAMME en SIMPLE PRECISION

(voir page suivante)

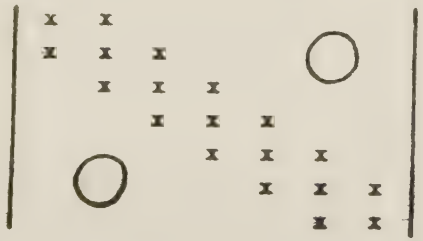
G E N E R A L I T E S

Définition d'une MATRICE-BANDE

On appelle "Matrice-Bande", toute matrice carrée d'ordre "n", dans laquelle tous les termes placés en dehors de la diagonale principale et d'un même nombre de diagonales parallèles, situées de part et d'autre de la diagonale principale, sont nuls; tous les autres termes étant à priori non nuls.

On désigne par " m " le nombre total de diagonales portant des termes non nuls. Le nombre " m " est un nombre toujours IMPAIR .

Ainsi par exemple, pour n = 7 et m = 3 , nous aurons, en représentant les termes non nuls par des " x ", une Matrice-Bande qui se présentera sous la forme suivante:



Organisation d'un Système Linéaire à Matrice-Bande.

Nous allons étudier l'organisation d'un système linéaire à Matrice-Bande, dont seuls les termes non nuls sont enregistrés, équation par équation, dans des mémoires consécutives à partir de l'adresse " M " .

Nous désignerons par " n " l'ordre du système linéaire à Matrice-Bande de caractéristique " m ", et par " p " le nombre de séries de seconds membres.

Soient " Adke " l'adresse du premier terme non nul du premier membre de la k° équation; et " Dake " l'adresse, augmentée de l'unité, du dernier second membre de la même équation. Les " n " équations étant enregistrées en mémoires les unes à la suite des autres, " Dake " représente aussi l'adresse du premier terme non nul du premier membre de la (k+1)° équation.

Pour k = 1, nous avons: Adke = M ; il suffit alors de connaître le nombre de termes non nuls composant la première équation pour déterminer immédiatement la valeur de Dake. Et ainsi de suite de proche en proche jusqu'à k = n, si l'on est capable de calculer le nombre de termes non nuls composant chacune des " n " équations.

Représentons le nombre de termes non nuls de la k° équation par:

$$Nbe + Nbt + p$$

avec :

Nbe : nombre de termes non nuls du premier membre ayant un rang inférieur à " k ".

Nbt : nombre de termes non nuls du premier membre ayant un rang égal ou supérieur à " k ".

p : nombre de seconds membres. Cette quantité est la même pour les " n " équations.

On établit rapidement les lois suivantes qui régissent les valeurs de " Nbe ", "Nbt" en fonction de celle de " k ".

1°) $1 \leqslant k \leqslant \frac{m+1}{2}$

Nous avons dans ce cas:

$$Nbe = k - 1$$

$$Nbt = \frac{m+1}{2}$$

2°) $\frac{m+1}{2} < k \leqslant n - \frac{m-1}{2}$

Nous avons:

$$Nbe = \frac{m-1}{2}$$

$$Nbt = \frac{m+1}{2}$$

Nous noterons que nous avons, dans ce second cas seulement:

$$Nbe + Nbt = m$$

3°) $n - \frac{m-1}{2} < k \leqslant n$

Dans ce dernier cas, nous avons:

$$Nbe = \frac{m-1}{2}$$

$$Nbt = n - k + 1$$

A titre d'exemple, on appliquera ces lois à la Matrice-Bande représentée au paragraphe précédent.

A partir de ces résultats, il est possible de calculer le nombre " N " de mémoires nécessaires à l'enregistrement des seuls termes non nuls d'un système linéaire à Matrice-Bande. En utilisant les mêmes notations que précédemment, on trouve:

$$N = (m + p) \times n - \frac{m^2-1}{4}$$

MEMOIRES de TRAVAIL

Toutes les notations utilisées sont celles définies au cours de l'étude détaillée du programme.

1°) <u>à contenus constants</u>.

(0082) = 69 (M) 0003
(0115) = 69 (M) 0053
(0136) = oo ($\frac{m+1}{2}$) oooo
(0155) = oo (p) oooo
(0193) = oo ($\frac{n-1}{2}$) oooo
(0199) = oo (n) oooo
(0231) = oo (m) oooo

2°) <u>à contenus variables</u>.

(0008) = Mot d'ORIENTATION
Son contenu sera utilisé pour aiguiller convenablement le programme à la SORTIE du Sous Programme d'ELIMI - NATION, pendant les calculs de la k° élimination qui intéresse les (k - 1) premières équations.
Nous pourrons trouver dans 0008:
10 0155 0209
44 0150 0042

(0014) = 69 xxxx 0053
Ce contenu sera utilisé pour l'appel des coefficients:
- soit de la k° équation au cours du Sous Programme de Travail sur la k° Equation.
Nous avons initialement: "xxxx" = adresse de la mémoire contenant le terme diagonal (de rang k) de la k° équation.
- soit des (k - 1) premières équations lors de l'élimination de leur k° inconnue.
Initialement, nous avons: (0014) = (0115)

(0020) = comptage des éliminations.
Le contenu de cette mémoire sera toujours négatif,

et initialement, nous avons:

(0020) = - oo oool oooo

nous faisons: k = 1 ($1 \leqslant k \leqslant n$).

(0024) = 24 xxxx 0060

Cette instruction participe au blocage de l'équation pivot à la suite des (k - 1) nouvelles équations obtenues après la k° élimination.

(0030) = oo (k) oooN

Ce contenu prendra toute sa signification à la fin de la Résolution de chaque Système Linéaire.
Nous dirons alors qu'au cours de l'élimination de la k° inconnue, nous avons eu une perte de précision de N chiffres. Cette perte est la perte maximum rencontrée.
La mémoire 0030 est remise à zéro au début de chaque résolution de système linéaire.

(0031) = oo (Adke) oooo

avec "Adke" = adresse de la mémoire contenant le premier terme du premier membre de la k° équation.

(0032) = 60 xxxx 0100

Le contenu de cette mémoire sera utilisé pour l'appel des coefficients de la k° équation qui seront divisés par le pivot.
Initialement, "xxxx" = adresse de la mémoire contenant le terme diagonal (de rang k) de la k° équation.

(0036) = oo (Nbt + p) oooo

Nous trouverons tout renseignement nécessaire dans le chapitre "GENERALITES", au paragraphe "Organisation d'un Système Linéaire à Matrice-Bande.

(0038) = oo xxxx oooo

Ce contenu représente le comptage:
- soit des éliminations (ou des inconnues éliminées) au cours du Sous Programme de Travail sur la k° Equation.
  Nous avons: $0000 \leqslant xxxx \leqslant Nbe$
- soit des équations, au cours du Sous Programme de la k° Elimination.
  Nous avons dans ce cas: $0000 \leqslant xxxx \leqslant (k - 1)$

(0044) = oo (Nbe) oooo

(0050) = 69 xxxx 0003

Cette instruction sera utilisée pour appeler successivement tous les termes de l'équation pivot convenable:
- lors de chacune des "Nbe" éliminations effectuées sur la k° équation.
  Nous aurons au début de la première de ces éliminations:
  xxxx = M + (Nbp + p)(k - 1 - Nbe)
  il s'agit de l'adresse du premier terme de la première équation pivot à utiliser.
- lors de la k° élimination portant sur les (k - 1) premières équations.
  Nous avons au début du traitement de chaque équation:
  xxxx = Adke + nbe
  il s'agit de l'adresse du terme de rang (k + 1) de l'équation pivot.

(0056) = perte maximum de précision.

Au début de la résolution de chaque système linéaire, cette mémoire est remise à zéro.

(0059) = oo (Adke + Nbe) oooo

En position facteur, nous avons: l'adresse de la mémoire contenant le terme diagonal de l'équation en cours de traitement.

(0064) = 21 xxxx 0203

Cette instruction sera utilisée pour:
- la mise en mémoire des résultats des "Nbe" éliminations effectuées sur la k° équation.
  Nous aurons, au début de chacune des "Nbe" éliminations:
  xxxx = Adke + Nbe
- le stockage des résultats de la k° élimination portant sur les (k - 1) premières équations.
  Nous aurons initialement:
  xxxx = M

(0070) = 60 oooo 0100 + (0059) + (0036)
En position facteur, nous avons alors: l'adresse, augmentée de l'unité, du dernier terme de la k° équation après l'élimination de ses (k - 1) premières inconnues (en fait les calculs d'élimination ne sont nécessaires que pour les "Nbe" dernières de celles-ci).
Le contenu de la mémoire 0070 sera utilisé pour tester la fin de la division de la k° équation par le pivot.

= 69 (Adep + Nbp + p) 0003
Ce contenu permet de tester la fin de chacune des "Nbe" éliminations effectuées sur la k° équation. "Adep" représente l'adresse du premier terme de l'équation pivot utilisée.

= 69 (Adke + Nbe + Nbt + p - 1) 0003
En position facteur, nous avons: l'adresse, augméntée de l'unité, du dernier second mémbre de la k° équation (équation pivot pour la k° élimination).
Le contenu de la mémoire 0070 est utilisé pour tester la fin de la k° élimination pour chacune des (k - 1) premières équations.

= 69 (Adke + Nbe + Nbt + p - 1) 8002
Une fois la k° élimination terminée, ce contenu servira à tester la fin du blocage de l'équation pivot (la k° équation) à la suite des (k - 1) nouvelles équations fournies par la k° élimination.

(0075) = Mot d'ORIENTATION
Il s'agit de la mémoire de SORTIE du Sous Programme d'Elimination.
Son contenu sera, lors de son utilisation:
- par l'intermédiaire du Sous Programme de Travail sur la k° Equation:
  10 0155 0109 puis
  44 0150 0072
- par l'intermédiaire du Sous Programme de la k° Elimination.
  10 0155 0209
  44 0150 0042

(0080) = Mot d'ORIENTATION
Il s'agit de la mémoire de SORTIE du Sous Programme de Division.
Son contenu sera:
  69 0140 0243 dans le cas de la division de la première équation par son terme de rang unité.
  60 0159 0144 dans tous les autres cas.

(0081) = 69 xxxx 8002
Cette instruction servira, à la fin de la k° élimi - nation, au blocage de l'équation pivot à la suite des (k - 1) nouvelles équations fournies par la k° élimination.
Initialement, nous aurons:
  xxxx = Adke + Nbe
il s'agit de l'adresse du terme de rang (k + 1) dans l'équation pivot.

(0094) = - oo (k - 1) oooo

(0103) = valeur exacte du pivot relatif à l'élimination de la k° inconnue, pendant le Sous Programme de DIVISION.

= multiplicateur, pour toutes les éliminations; c'est à dire:
- soit pendant le Sous Programme de Travail sur la k° Equation.
- soit pendant le Sous Programme de la k° Elimination.

(0110) = instruction de SORTIE du Sous Programme de Résolution des Systèmes Linéaires à Matrice-Bande.

(0114) = 21 xxxx 0207
Cette instruction permettra la mise en mémoire des quotients obtenus en divisant les termes de la k° équation par le pivot (le terme de rang k de la k° équation).
Initialement, nous avons:
  xxxx = Adke + Nbe
il s'agit de l'adresse du terme de rang k de la k° équation.

(0122) = 69 xxxx 0058
Instruction utilisée au cours du Sous Programme de

la k^{e} élimination, pour l'appel du multiplicateur relatif à l'équation que nous nous apprêtons à traiter. Nous aurons toujours: "xxxx" égal à l'adresse du terme de rang k dans cette équation.

(0123) = 21 xxxx 0150

Instruction participant à la préparation du calcul du terme de rang (k + Nbp) (ceci n'est à considérer que pour $k + Nbp \leqslant n$) dans chacune des (k - 1) nouvelles équations fournies par la k^{e} élimination.

(0159) = oo (Nbp + p) oooo

(0174) = oo (Dake) oooo

"Dake" représente l'adresse, augmentée de l'unité, du dernier second membre de l'équation en cours de traitement. Initialement, il s'agit de la première équation, nous avons:

$$Dake = M + \frac{m+1}{2} + p$$

(0204) = Mot d'ORIENTATION

Son contenu est utilisé pour aiguiller convenablement le programme à la SORTIE du Sous Programme d'Elimi - nation, pendant les calculs des "Nbe" éliminations qui intéressent la k^{e} équation.

Nous pourrons trouver dans 0204:

1C 0155 0109

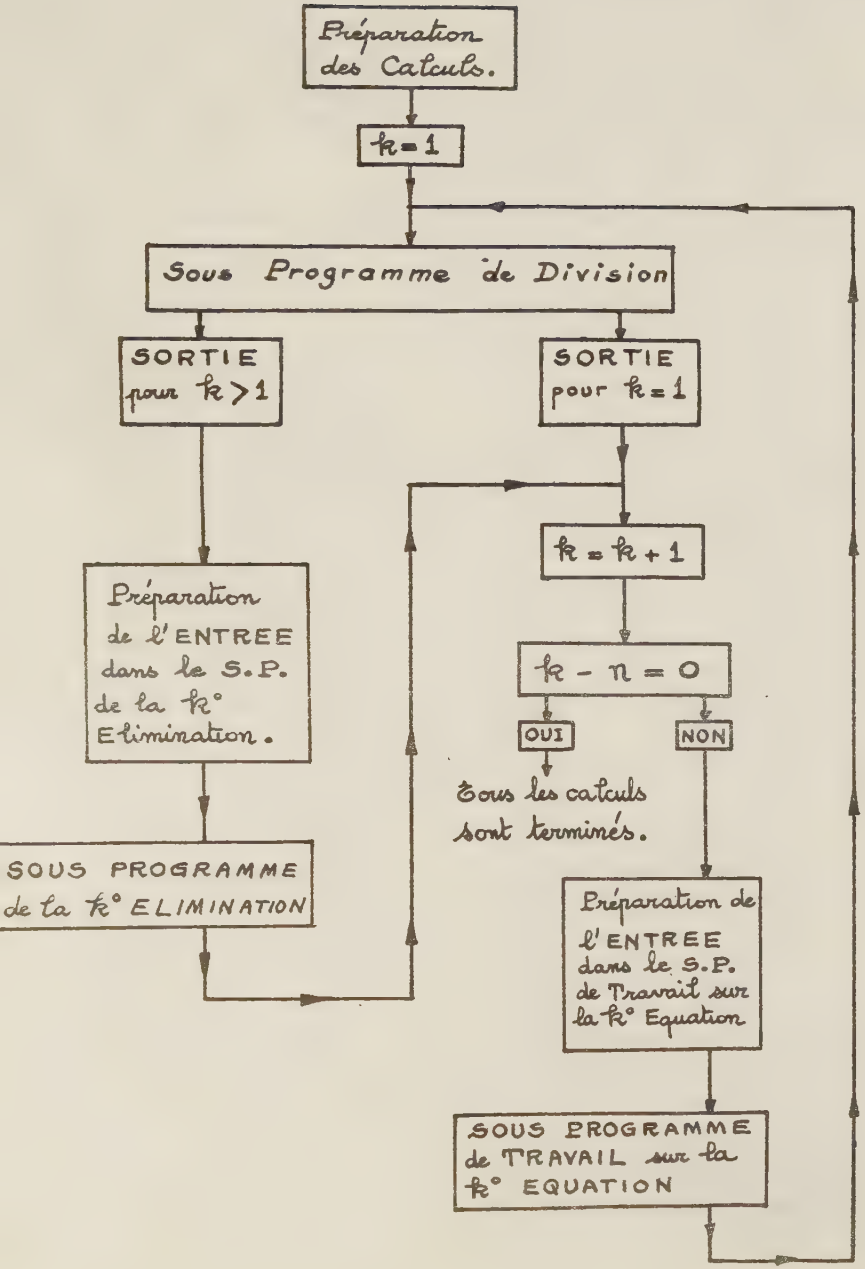
44 0150 0072

(1953) = perte de précision relative à la dernière opération effectuée en Arithmétique Virgule Flottante.

PROGRAMME DE LIAISON

OBJET

Ce Programme réalise, à l'aide de Sous Programmes convenables, la chaîne de calculs reptésentés par l'organigramme ci-après.



Nous appellerons " n " l'ordre du système à résoudre, et " p " le nombre de vecteurs seconds membres. La Matrice-Bande du système sera caractérisée par le nombre de diagonales non nulles que nous appellerons: " m ".

Les données sont enregistrées sur le tambour à partir d'une mémoire d'adresse " M ". LE système est rangé équation par équation. Chaque équation comprend les termes non nuls du premier membres (au maximum "m" termes) et les "p" seconds membres.

Pour entrer dans le Sous Programme de Résolution de Systèmes Linéaires à Matrice-Bande, il faut charger 8001, 8002, 8003 de la façon suivante:

(8001) = une instruction de SORTIE.
(8002) = (m) (p) oooo
(8003) = (M) (n) oo

La première instruction de ce Sous Programme se trouve en 0000.

Initialisation

0000	24 0110 0015	(0110) = une instruction de SORTIE
0015	69 0022 0226	(0022) = oo oool oooo
0226	22 0155 0160	(0155) = oo (p) oooo
0160	30 0004 0172	
0172	22 0231 0134	(0231) = oo (m) oooo
0134	30 0004 0045	
0045	22 0199 0002	(0199) = oo (n) oooo
0002	30 0004 0065	
0065	22 0059 0162	(0059) = oo (M) oooo

M = adresse du terme diagonal de la première équation.

Par la suite, nous aurons

toujours en position facteur du contenu de la mémoire 0059 : l'adresse du terme diagonal de l'équation en cours de traitement; c'est à dire, en utilisant les notations définies précédemment : Adke + Nbe

0162	69 0115 0222	(0115) = 69 oooo 0053
0222	22 0115 0223	(0115) = 69 (M) 0053
0223	69 0082 0185	(0082) = 69 oooo 0003
0185	22 0082 0235	(0082) = 69 (M) 0003
0235	66 0022 0132	(0022) = oo oool oooo
0132	20 0020 0074	(0020) = - oo oool oooo

Comptage des éliminations, $k = 1$

0074	21 0056 0210	(0056) = - oo oooo oooo
0210	24 0030 0202	(0030) = - oo oooo oooo
0202	15 0231 0086	(0231) = oo (m) oooo
0086	14 0139 0249	(0139) = oo oooo ooo2
0249	10 8002 0212	
0212	10 0022 0182	(0022)
0182	21 0136 0189	(0136) = oo ($\frac{m+1}{2}$) oooo
0189	20 0193 0096	(0193) = oo ($\frac{m-1}{2}$) oooo
0096	15 0155 0165	(0155) = oo (p) oooo
0165	10 8001 0124	
0124	21 0036 0239	(0036) = oo (Nbt + p) oooo

initialement, nous avons : $Nbt + p = \frac{m+1}{2} + p$

0239	20 0159 0215	(0159) = oo (Nbp + p) oooo

initialement, nous avons : $Nbp + p = \frac{m-1}{2} + p$

0215	10 0059 0166	(0059) = oo (Adke + Nbe) oooo

initialement, nous avons : Adke + Nbe = M

0166	21 0174 0232	(0174) = oo (Dake) oooo

Dake = adresse, augmentée de l'unité, du dernier second membre de l'équation en cours de traitement.

Initialement, il s'agit de la première équation; nous avons: $Dake = M + \frac{m+1}{2} + p$

0232	69 0186 0090	(0186) = 69 0140 0243
0090	24 0080 0156	(0080)

Préparation de la SORTIE du Sous Programme de Division.

Nous avons à ce moment:

$$(0059) = oo\ (\ M\ )\ oooo$$
$$(0036) = oo\ (\ \frac{m+1}{2} + p)\ oooo$$

Nous commençons la résolution du système linéaire. La première équation servira d'équation pivot pour l'élimination de la 1° inconnue.
Nous allons diviser cette première équation, composée de $Nbt + p = \frac{m+1}{2} + p$ termes non nuls, par son terme diagonal, c'est à dire: le pivot.
Nous obtiendrons alors une équation pivot composée de $\frac{m-1}{2} + p = Nbp + p$ termes non nuls et différents de l'unité.
Ensuite, nous serons prêts à aborder le "traitement" de la 2° équation. Et à partir de ce moment, tous les renseignements joints aux divers Sous Programmes seront applicables en faisant k = 2 .

0156

Sous Programme de DIVISION

0080

0080	69 0140 0243	(0140) = 60 0159 0144
0243	24 0080 0181	(0080)

Préparation de la SORTIE du Sous Programme de Division.
Cette SORTIE restera inchangée jusqu'à la fin des calculs.

Test de fin de résolution du système linéaire.

(10)	0181	60 0020 0083	(0020) = - oo (k) oooo
	0083	24 0094 0047	(0094)
	0047	10 0199 0216	(0199) = oo (n) oooo
	0216	44 0224 0110	

Les "n" inconnues n'ont pas toutes été éliminées.
Nous allons commencer l'élimination de l'inconnue suivante.

Détermination des valeurs de "Nbe", "Nbt+p" .
Préparation de l'ENTREE dans le Sous Programme de Travail sur la K° équation.

0224	11 8001 0133	
0133	11 0022 0183	(0022) = oo oool oooo
0183	21 0020 0233	(0020) = - oo (k) oooo
		(0094) = - oo (k-1) oooo
0233	10 0136 0092	(0136) = oo ($\frac{m+1}{2}$) oooo
0092	46 0095 0146	

Nous avons: $1 < k \leqslant \frac{m+1}{2}$

Nous devons avoir: Nbt+p = $\frac{m+1}{2}$ + p,

il n'y aura donc pas lieu de modifier le contenu de la mémoire 0036 ; et d'autre part: Nbe = k - 1 .

0146	61 0094 0151	(0094)
0151	21 0044 0097	(0044) = oo (Nbe) oooo
		Nbe = k-1
0097	Voir à la page suivante en (11).	

Nous avons: $k > \frac{m+1}{2}$

Dans ce cas, nous devons avoir

Nbe = $\frac{m-1}{2}$

0095	69 0193 0196	(0193) = oo ($\frac{m-1}{2}$) oooo
0196	24 0044 0147	(0044) = oo (Nbe) oooo
		Nbe = $\frac{m-1}{2}$
0147	10 0199 0184	(0199) = oo (n) oooo
0184	11 0231 0236	(0231) = oo (m) oooo
0236	46 0190 0097	(8003) = oo ($-k+\frac{m+1}{2}+n-m$) oooo

Nous avons: $n - \frac{m-1}{2} < k \leqslant n$

Dans ce cas, la quantité (Nbt + p) décroit de une unité à chaque fois que "k" augmente de la même quantité, et ceci à partir de sa valeur initiale: $\frac{m+1}{2}$ + p .

Dans ce cas aussi,(voir la note explicative accompagnant le Sous Programme de Travail sur la k° Equation) nous devons charger la mémoire de SORTIE du Sous Programme d'Elimination avec : 44 0150 0072

0190	60 0036 0142	(0036) = oo (Nbt + p) oooo
0142	11 0022 0234	(0022) = oo oool oooo
0234	21 0036 0240	(0036)
0240	69 0018 0087	(0018) = 44 0150 0072
0087	Voir quelques lignes plus bas en (12)	

Nous avons: $1 < k \leqslant n - \frac{m-1}{2}$

Les valeurs de "Nbe" et de "Nbt+p" ont déjà été déterminées, il nous reste à charger convenablement la mémoire de SORTIE du Sous Programme d'Elimination. En nous référant à la note explicative jointe au Sous Programme de Travail sur la k° Equation, nous voyons qu'il faut y charger: 10 0155 0109.

(11)	0097	69 0001 0087	(0001) = 10 0155 0109
(12)	0087	24 0204 0137	(0204)

Le contenu de la mémoire 0204 sera placé en 0075 (mémoire de SORTIE du S. P. d'Elimination) au début du S.P. de Travail sur la k° Equation .

```
0137    65 0174 0187      (0174) = oo (Dake) oooo
                          Dake égale, en ce moment,
                          l'adresse,augmentée de l'unité,
                          du dernier second membre de la
                          (k-1)°équation.
                          Cette adresse est donc égale
                          à celle du premier terme du
                          premier membre de la k° équation
                          c'est à dire Adke.
0187    24 0031 0237      (0031) = oo (Adke) oooo
0237    15 0044 0201      (0044) = oo ( Nbe ) oooo
0201    20 0059 0088      (0059) = oo (Adke+Nbe) oooo
                          en position facteur, nous avons
                          l'adresse du terme de rang k
                          (terme diagonal) dans le premier
                          membre de la k° équation.
0088    21 0038 0192,     (0038) = oo oooo oooo
0192    15 0036 0242      (0036) = oo (Nbt + p) oooo
0242    20 0174 0138      (0174) = oo (Dake) oooo
                          Dake égale maintenant l'adresse,
                          augmentée de l'unité, du dernier
                          second membre de la k° équation.
0138    61 0094 0052      (0094) = - oo (k-1) oooo
0052    11 0044 0102      (0044)
0102    19 0159 0188      (0159) = oo (Nbp + p) oooo
0188    30 0004 0152
0152    15 0082 0238      (0082) = 69 ( M ) 0003
0238    22 0050 0206      (0050) = 69 Adep 0003
                          Adep = M + (Nbp + p)(k - 1 - Nbe)
                          La signification de cette
                          adresse est donnée dans la note
                          explicative jointe au S.P. de
                          Travail sur la k° Equation.
0206    15 0159 0063      (0159)
0063    20 0070 0200      (0070) = 69 (Adep + Nbp + p)
                                                  0003
```

0200
Sous Programme de TRAVAIL sur la k° EQUATION
0156

En ce moment, la k° équation est constituée de la façon suivante:

- les termes, du premier membre, de rang inférieur à k sont tous nuls.
- les termes de rangs: k, k+1, k+2, ... , k+Nbt-1 sont quelconques.
- ensuite, jusqu'au rang "n" inclus, les termes sont tous nuls.
- finalement, nous avons "p" seconds membres.

Seuls les termes non nuls sont conservés en mémoires. Nous les trouvons dans (Nbt + p) mémoires consécutives à partir de l'adresse (Adke + Nbe).

La k° équation va maintenant être prise comme équation pivot pour l'élimination de la k° inconnue. Le pivot sera donc le terme de rang "k" (terme diagonal) de la k° équation.

Nous allons entrer dans le Sous Programme de Division, pour diviser les termes non nuls de la k° équation par le pivot. Le diviseur se trouve dans la mémoire d'adresse (Adke + Nbe); les (Nbt + p - 1) dividendes sont dans des mémoires consécutives à partir de l'adresse (Adke + Nbe + 1) . Les (Nbt + p - 1) quotients seront conservés dans des mémoires consécutives à partir de l'adresse (Adke + Nbe).

0156
Sous Programme de DIVISION
0080

Nous avons alors:
(0080) = (0140)

Préparation de l'ENTREE dans le Sous Programme de la k° Elimination

```
0080    60 0159 0144      (0159) = oo (Nbp + p) oooo
0144    11 0036 0194      (0036) = oo (Nbt + p) oooo
0194    44 0197 0048
```

Nous avons: Nbp≠Nbt ; c'est à dire: $1 < k \leqslant n - \frac{m-1}{3}$

Comme on le verra dans la note explicative jointe au Sous Programme de la k° Elimination, nous devons dans ce cas charger la mémoire 0008 avec: 10 0155 0209.

0197 69 0101 0244 (0101) = 10 0155 0209
0244 Voir quelques lignes plus bas en (13)

Nous avons; Nbp = Nbt ; c'est à dire: $n - \frac{m-1}{2} < k \leqslant n$

Comme on le verra dans la note explicative jointe au Sous Programme de la k° Elimination, nous devons dans ce cas charger la mémoire 0008 avec: 44 0150 0042.

D'autre part, à partir de la première fois que nous rencontrons: $k > n - \frac{m-1}{2}$, la quantité (Nbp + p) diminue d'une unité chaque fois que "k" augmente de la même quantité. C'est ici que nous allons effectuer cette opération en prévision de la prochaine élimination de la (k+1)° inconnue. (elle commencera dès la SORTIE DU Sous PRogramme de la k° Elimination en 0181).

0048 10 8001 0145
0145 11 0022 0195 (0022) = oo oool oooo
0195 21 0159 0245 (0159) = oo (Nbp + p) oooo
nouvelle valeur
0245 69 0037 0244 (0037) = 44 0150 0042
(13) 0244 24 0008 0246 (0008)
Le contenu de la mémoire 0008 sera utilisé pour aiguiller convenablement le programme à la SORTIE du SOUS PROGRAMME d'ELIMINATION.
0246 65 0115 0247 (0115) = 69 (M) 0053
0247 24 0014 0098 (0014)
0098 69 0064 0148 (0064) = 21 xxxx 0203
0148 22 0064 0198 (0064) = 21 (M) 0203
0198 21 0038 0051 (0038) = oo oooo oooo

0051
Sous Programme de la k° ELIMINATION
0181

0181 Voir en (10) dans la 5° page précédente.

Les "n" inconnues ont été éliminées. La résolution du système linéaire est terminée.

0110 Intruction de SORTIE.

Les racines correspondant aux "p" séries de seconds membres occupent un bloc de p n mémoires consécutives à partir de l'adresse " M " . Les racines sont disposées de telle façon que nous trouverons celles correspondantes à la j° série de seconds membres dans les mémoires d'adresses:

M + j - 1
M + j - 1 + p
M + j - 1 + 2p
.....
M + j - 1 + (n-1)p

D'autre part le contenu de la mémoire 0030, nous donne la perte maximum de précision rencontrée au cours de la résolution du système linéaire.

(0030) = oo (k') (Perte maximum de précision)

la valeur de k' permet de situer à quel moment nous avons rencontré la perte maximum de précision, elle indique quelle inconnue était en cours d'élimination.

La machine est prête à résoudre un nouveau système linéaire.

ETUDE DES SOUS PROGRAMMES

SOUS PROGRAMME DE DIVISION

OBJET

Au cours de chaque élimination, ce Sous Programme divise l'équation pivot par le pivot.

L'adresse du pivot se trouve en position facteur du contenu de la mémoire 0059.

Tous les termes de l'équation se trouvent en séquence à la suite du pivot.

Le nombre de termes de l'équation pivot se trouve en position facteur du contenu de la mémoire 0036 . Ce même nombre diminué de l'unité, indique le nombre de divisions à effectuer.

La mise en mémoires des résultats se fait à partir de celle occupée par pivot.

ENTREE

Il suufit de charger convenablement les mémoires 0059 et 0036 .

Les contenus de 8003, 8002, 8001 peuvent être quelconques.

SORTIE

Elle est sous la dépendance du Programme de LIAISON .

```
0156   65 0059 0113        (0059)
0113   15 0066 8002        (0066) = 60 oooo 0107
8002   60 xxxx 0107
0107   44 0012 0062
```

LE pivot est nul.

```
0062   01 9999 9999
```

Le pivot n'est pas nul.

```
      0012    24 0103 0157        (0103) = valeur exacte du
                                          pivot,
                                          qui sera le diviseur.
      0157    65 0059 0163        (0059)
      0163    69 0114 0068        (0114) = 21 oooo 0207
      0068    22 0114 0118
      0118    15 0121 0175        (0121) = 60 oooo 0100
      0175    10 8002 0033
      0033    15 0036 0091        (0036)
      0091    20 0070 0023        (0070)
(1)   0023    10 0022 0177        (0022) = oo oool oooo
      0177    21 0032 0135        (0032)
      0135    11 0070 0225
      0225    44 0032 0080
```

Les calculs ne sont pas terminés. Nous allons effectuer la division suivante.

```
      0032    60 xxxx 0100
              Séquence d'ENTREE en A.V.F. pour effectuer une
              Division.
```

```
0100    69 0103 0006        (0103) = Diviseur.
0006    24 1801 0104
0104    69 1900 0153        VDV
0153    24 1808 0111
0111    69 0114 1800
```

```
1800
   Sous-Programme
      A. V. F.
                  1803      (1803) = (0114)
                                   = 21 xxxx 0207
```

Mise en mémoire du résultat.

```
      1803    21 xxxx 0207
      0207    65 0114 0169        (0114) = 21 xxxx 0207
      0169    15 0022 0227        (0022) = oo oool oooo
      0227    20 0114 0168
      0168    60 0032 0023  (1)   (0032) = 60 xxxx 0100
```

Toutes les Divisions sont terminées.

```
      0080    SORTIE du Sous Programme de DIVISION.
```

SOUS PROGRAMME d'ELIMINATION

OBJET

Soient trois nombres A, B et C en virgule flottante FLAIR . Les nombres B ET C sont dans des mémoires dont les adresses variables sont respectivement indiquées par les facteurs des contenus des mémoires 0050 et 0014 . Le nombre A, qui peut être le même pour toute une série de calculs, se trouve dans la mémoire 0103 .

Ce Sous Programme effectue, à l'aide de l'A.V.F. , l'opération-A×B + C, puis met le résultat dans une mémoire d'adresse variable indiquée par le facteur du contenu de la mémoire 0064, et finalement prépare l'opération suivante.

De plus, ce Sous Programme détecte la perte maximum de précision rencontrée au cours des calculs .

ENTREE

Indifférente.

SORTIE

Sous la dépendance du Programme de LIAISON .

```
0150    61 0103 0050        (0103) = mutiplicateur.
```

Séquence d'ENTREE en A.V.F. pour multiplier A par B .

```
0050    69 xxxx 0003
0003    24 1801 0004
0004    69 1996 0049        VMU
0049    24 1808 0011
0011    69 0014 1800
```

```
1800
   Sous-Programme
      A. V. F.
                1803
```

(1803) = (0014)
= 69 xxxx 0053

Séquence d'ENTREE dans l'A.V.F. pour additionner C au résultat de la multiplication précédente.

```
1803    69 xxxx 0053
0053    24 1801 0054
0054    69 1902 0005        VAA
0005    24 1808 0061
0061    69 0064 1800
```

```
1800
   Sous-Programme
      A. V. F.
                1803
```

(1803) = (0064)
= 21 xxxx 0203

Mise en mémoire du résultat.

```
1803    21 xxxx 0203
```

Etude de la perte de précision.

```
0203    60 0056 0161        (0056) = perte maximum rencontrée
                                     depuis le début des
                                     calculs.
0161    11 1853 0007        (1853) = perte de précision
                                     lors de la dernière
                                     opération.
0007    46 0010 0211
```

La perte de précision de la dernière opération est supérieure aux pertes antérieures.

```
0010    24 0056 0009        (0056)
0009    60 8001 0017
```

```
0017    11 0020 0025        (0020) = - oo ( k ) oooo
                            k = numèro d'ordre de l'élimi-
                                nation. 1 ⩽ k ⩽ n
0025    21 0030 0211        (0030)
```

Préparation de l'opération suivante.

```
0211    65 0064 0019        (0064) = 21 xxxx 0203
0019    15 0022 0027        (0022) = oo oool oooo
0027    20 0064 0067
0067    60 0050 0055        (0050) = 69 xxxx 0003
0055    15 0014 0069        (0014) = 69 xxxx 0053
0069    10 0022 0077
0077    15 8001 0035
0035    21 0050 0154
0154    20 0014 0117
(6) 0117    11 0070 0075    (0070) = 69 yyyy 0003
                            La quantité représentée
                            par yyyy est sous la
                            dépendance d'autres
                            Sous Programmes.
0075    SORTIE du SOUS PROGRAMME d'ELIMINATION.
```

SOUS PROGRAMME DE TRAVAIL SUR LA k° EQUATION

OBJET

Quand nous commençons le traitement de la k° équation, ($k \neq 1$), les (k-1) premières inconnues ont déjà été éliminées.

La k° équation occupe des mémoires consécutives à partir d'une adresse que nous appellerons: "Adke". Il s'agit d'éliminer, dans cette équation, toutes les inconnues de rang inférieur à k.

Le premier membre de la k° équation se compose de :

- "Nbe" termes de rang inférieur à k ; soit par conséquent (k-1), (k-2), ... ,(k-Nbe). Ce sont les termes à éliminer. Ils seront à utiliser comme multiplicateurs au cours des "Nbe" éliminations à effectuer sur la k° équation.
  Nous aurons:

$$Nbe = k-1 \quad \text{pour} \quad 1 < k \leqslant \frac{m+1}{2}$$

$$Nbe = \frac{m-1}{2} \quad \text{pour} \quad \frac{m+1}{2} < k \leqslant n$$

- "Nbt" termes de rang supérieur et égal à k ; soit par conséquent: k, (k+1), ... , (k+Nbt-1) Ce sont sur ces termes, ainsi que sur les "p" seconds membres, que porteront les calculs d'élimination.
  Nous aurons:

$$Nbt = \frac{m+1}{2} \quad \text{pour} \quad 1 < k \leqslant n - \frac{m-1}{2}$$

$$Nbt = n - k + 1 \quad \text{pour} \quad n - \frac{m-1}{2} < k \leqslant n$$

Les (k-1) premières équations sont les équations pivots à utiliser pour l'élimination des (k-1) premières inconnues. Elles sont toutes construites de la même façon, comme nous le verrons lors de l'étude du SOUS PROGRAMME de la k° ELIMINATION. Ainsi par exemple, si nous considérons la j° équation:

- tous les termes de rang inférieur à k sont nuls sauf celui de rang j , qui est égal à l'unité.
- à partir du rang k, nous trouvons "Nbp" termes non nuls, puis à nouveau des termes nuls jusqu'au rang n inclus.
- enfin "p" seconds membres terminent la j° équation.

Nous conservons en mémoires (Nbp + p) termes par équation. Les (k-1) équations ainsi réduites sont enregistrées en mémoires les unes à la suite des autres, dans des mémoires consécutives à partir de l'adresse "M".

Pour nos "Nbe" éliminations, il nous faut les "Nbe" dernières équations pivots. Par conséquent, l'adresse du premier terme de la première équation pivot à utiliser, sera :

$$Adep = M + (k - 1 - Nbe)(Nbp + p)$$

En ce qui concerne "Nbp", nous aurons:

$Nbp = Nbt - 1$ pour $1 < k \leqslant n - \frac{m-1}{2}$

$Nbp = Nbt$ pour $n - \frac{m-1}{2} < k \leqslant n$

REmarque

Dans le premier cas (Nbp = Nbt - 1) le terme de rang (k + Nbt - 1) de la k° équation restera inchangé au cours des "Nbe" éliminations.

ENTREE dans le Sous Programme de Travail sur la k° Equation.

Huit mémoires devront être convenablement chargées. Il faudra en effet vérifier:

(0031) = oo Adke oooo
(0038) = oo oooo oooo
(0044) = oo (Nbe) oooo
(0050) = 69 Adep 0003
(0059) = oo (Adke + Nbe) oooo
(0070) = 69 (Adep + Nbp + p) 0003
(0159) = oo (Nbp + p) oooo

(0204) = 10 0155 0109
dans le cas où Nbp = Nbt - 1
= 44 0150 0072
dans le cas où Nbp = Nbt.

Les contenus de 8003, 8002, 8001 peuvent être quelconques.

SORTIE

Nous quitterons toujours le "Sous Programme de Travail sur la k° Equation" pour entrer dans le "Sous Programme de Division".

Préparation de la SORTIE du Sous Programme d'Elimination

(4)	0200	69 0204 0057	(0204)
	0057	24 0075 0028	(0075)

Appel et mise en mémoire du multiplicateur pour la prochaine élimination.

	0028	65 0031 0085	(0031) = oo Adke oooo
	0085	15 0038 0043	(0038) = compteur des éliminations.
	0043	15 0046 8002	(0046) = 69 oooo 0105
	8002	69 xxxx 0105	
	0105	24 0103 0106	(0103) = multiplicateur
	0106	65 0059 0013	(0059) = oo(Adke + Nbe) oooo
	0013	15 oo16 0021	(0016) = 69 oooo 0053
(2)	0021	20 0014 0167	(0014)
	0167	69 0064 0217	(0064) = 21 xxxx 0203
	0217	22 0064 0150	(0064) = Instruction pour la mise en mémoires des résultats. Les termes de la k° équation conserve la même place au cours des "Nbe" éliminations.

(3) 0150
Sous Programme d' ELIMINATION
0075

Nous avons à ce moment:
(0075) = (0204)

Il faut distinguer deux cas:

1°) Nous avons Nbp = Nbt - 1

Il nous faut détecter la fin du premier membre de l'équation pivot. Ainsi, nous pourrons éviter tout calcul sur le terme de rang (k + Nbt - 1) de la k° équation.

```
0075   10 0155 0109      (0155) = oo ( p ) oooo
                         (8003) = (0050) - [(0070)-(0155)]
                                =   69 xxxx 0003
                                  - 69 (Adep + Nbp) 0003
0109   44 0150 0164
```

Nous n'avons pas utilisé tous les termes du premier membre de l'équation pivot.

```
0150                (3)
```

Nous avons utilisé tous les termes du premier membre de l'équation pivot.

```
0164   69 0018 0071      (0018) = 44 0150 0072
0071   24 0075 0078      (0075)
                         Modification de la SORTIE du
                         Sous Programme d'Elimination,
                         pour les calculs avec les
                         seconds membres.
0078   65 0014 0119      (0014) = 69 xxxx 0053
                         Nous avons en position facteur,
                         l'adresse du terme de rang
                         (k+Nbt-1) dans le premier
                         membre de la k° équation.
0119   15 0022 0021      (0022) = oo oool oooo
                         En position facteur de 8002,
                         nous avons alors l'adresse du
                         premier second membre de la
                         k° équation.
0021                (2)
```

2°) Nous avons Nbp = Nbt.

ou bien, nous travaillons sur les seconds membres de la k° équation dans le cas où: Nbp = Nbt - 1 .

```
0075   44 0150 0072      (8003) = (0050) - (0070)
                                =   69 xxxx 0003
                                  - 69 (Adep+Nbp+p) 0003
```

L'élimination en cours n'est pas terminée.

```
0150                (3)
```

L'élimination en cours est terminée.

```
0072   60 0038 0093      (0038) = compteur des
                                  éliminations.
0093   10 0022 0127      (0022) = oo oool oooo
0127   21 0038 0041
0041   11 0044 0099      (0044) = oo (Nbe) oooo
0099   44 0205 0156
```

Les "Nbe" éliminations ne sont pas terminées.
Nous allons préparer l'élimination suivante. La première adresse de la nouvelle équation pivot se trouve en position facteur du contenu de 0070. Cette nouvelle équation pivot compte (Nbp + p) termes. Les (Nbt + p) termes de la k° équation se trouvent toujours dans les mêmes mémoires consécutives à partir de l'adresse (Adke + Nbe).

```
0205   65 0070 0125      (0070)
0125   24 0050 0206      (0050)
0206   15 0159 0063      (0159) = oo (Nbp + p) oooo
0063   20 0070 0200      (0070)
0200                (4)
```

Les "Nbe" éliminations sont terminées.

```
0156   SORTIE du Sous Programme de Travail sur la
       k° Equation.
```

SOUS PROGRAMME de la k° ELIMINATION

OBJET

Ce Sous Programme a pour but d'éliminer la k° inconnue dans les (k-1) premières équations; la k° équation sera l'équation pivot. Le pivot est donc pris obligatoirement sur la diagonale principale de la Matrice-Bande du système.

La k° équation est, à ce moment là, constituée de la façon suivante:

- tous les termes de rang inférieur à k sont nuls.
- le terme de rang k est égal à l'unité.
- jusqu'au rang (k + Nbt - 1) nous avons des termes non nuls; puis des termes nuls jusqu'au rang n inclus.
- enfin "p" seconds membres complètent cette k° équation.

Seuls les (Nbt - 1 + p) termes non nuls, ou différents de l'unité, sont conservés dans des mémoires consécutives à partir de l'adresse (Adke + Nbe).

La k° élimination portera sur (k-1) équations rangées les unes à la suite des autres dans des mémoires consécutives, à partir de l'adresse "M". Chacune de ces équations se compose de (Nbp + p) termes non nuls, qui se répartissent comme suit:

- Nbp termes de rangs: k, k+1, k+2, ... , k+Nbp-1.
- "p" seconds membres.

Les calculs d'élimination sont les mêmes pour toutes les équations. Ainsi de la j° équation (j = 1, 2, 3, ..., (k-1)) nous retrancherons : l'équation pivot multipliée par le terme de rang k de la j) équation. De cette façon, nous obtiendrons (k-1) nouvelles équations identiques. Le modèle de ces nouvelles équations dépend des valeurs des quantités "Nbp" et "Nbt" définies précédemment. Il n'y a que deux cas possibles :

1°) $Nbp = Nbt - 1$, pour $1 < k \leqslant n - \frac{m-1}{2}$

En ne considérant que les termes de rang supérieur à k ; nous constatons que la k° équation (équation pivot) ne diffère des (k-1) premières équations que par la présence , dans son premier membre, d'un terme de rang $(k+Nbt-1)$ non nul. En conséquence, à la fin des calculs d'élimination de la k° inconnue, nous aurons (k-1) nouvelles équations composées de (Nbp + p) termes non nuls se répartissant comme suit:

- "Nbp" termes de rangs: k+1, k+2, k+3, ... , k+Nbp.
- "p" seconds membres.

La k° équation (équation pivot) répond au même modèle puisque nous avons:

$$k+Nbp = k+Nbt-1.$$

Remarque:

Le terme de rang (k+Nbp) du premier membre d'une des (k-1) nouvelles équations est égal au produit, changé de signe, du terme de rang (k+Nbt-1) du premier membre de l'équation pivot par le terme de rang k du premier membre de l'équation correspondante parmi les (k-1) équations initiales.

2°) $Nbp = Nbt$ pour $n - \frac{m-1}{2} < k \leqslant n$

En ne considérant que les termes de rang supérieur à k ; nous constatons que la k° équation (équation pivot) et les (k-1) premières équations sont constituées de la même façon. En conséquence, à la fin des calculs d'élimination de la k° inconnue, nous aurons (k-1) nouvelles équations composées de (Nbp - 1 + p) termes non nuls répartis de la façon suivante :

- "Nbp-1" termes de rangs: k+1, k+2, ... , k+Nbp-1.

- "p" seconds membres.

La k° équation (équation pivot) répond au même modèle puisque nous avons:

$$k+Nbp-1 = k+Nbt-1.$$

Remarque

Dans ce cas, nous avons: Nbp = Nbt = n-k+1 . Donc , Nbt et Nbp diminuent lorsque k augmente.

Dans les deux cas exposés ci-dessus, les (k-1) nouvelles équations sont enregistrées en mémoires les unes à la suite des autres, dans des mémoires consécutives à partir de l'adresse "M". Une fois tous les calculs terminés; nous bloquerons à la suite de ces équations: la k° équation qui leur est identique.

En opérant de cette façon, lorsque nous aborderons le traitement de la (k+1)° équation, nous nous retrouverons dans la même situation qu'au début du traitement de la k° équation. Il suffira de faire k = k+1 , et ainsi de suite jusqu'à k =n.

Après l'élimination de la n° inconnue, nous disposerons de n équations composées chacune de (Nbp-1+p) termes non nuls ou différents de l'unité. Mais nous avons aussi:

Nbp = n-k+1 donc pour k=n, Nbp = n-n+1 = 1

par suite: (Nbp-1+p) est dans ce cas égal à "p". Ce qui signifie que nous aurons alors les "p" séries de "n" racines bloquées en mémoires à partir de l'adresse "M". Nous trouverons la r° série de racines dans les mémoires d'adresses:

M + r - 1
M + r - 1 + p
M + r - 1 + 2p
.....
M + r - 1 + (n-1)p

ENTREE dans le Sous Programme de la k° Elimination

En admettant les symboles déjà définis, six mémoires devront être chargées de la façon suivante:

(0014) = 69 (M) 0053
(0036) = oo (Nbt + p) oooo
(0038) = oo oooo oooo
(0059) = oo (Adke + Nbe) oooo
(0064) = 21 (M) 0203
(0008) = 10 0155 0209
dans le cas où Nbp = Nbt - 1
= 44 0150 0042
dans le cas où Nbp = Nbt.

Les contenus de 8003, 8002, 8001 peuvent être quelconques.

SORTIE

Nous quitterons le Sous Programme de la k° Elimination pour retourner dans le Programme de LIAISON.

Préparation de la SORTIE du Sous Programme d'Elimination.

(5)	0051	69 0008 0112	(0008)
	0112	24 0075 0128	(0075)

Appel et mise en mémoire du multiplicateur pour la prochaine élimination.

0128	65 0014 0219	(0014) = 69 xxxx 0053 en position facteur, nous avons initialement: M .
0219	69 0122 0026	(0122) = 69 xxxx 0058
0026	22 0122 8001	
8001	69 xxxx 0058	

0058	24 0103 0108	(0103) = multiplicateur.
0108	15 0022 0178	(0022) = oo oool oooo

Préparation des calculs sur une équation.

0178	20 0014 0218	(0014) = 69 ▪▪▪▪ 0053 en position facteur, nous avons l'adresse du terme de rang k+1
0218	60 0059 0213	(0059) = oo (Adke + Nbe) oooo
0213	1o 0116 0171	(0116) = 69 oooo 0003
0171	21 0050 0158	(0050) = 69 (Adke+Nbe) 0003 en position facteur, nous avons l'adresse du terme de rang k+1 de l'équation pivot.
0158	10 0036 0141	(0036) = oo (Nbt+p) oooo
0141	11 0022 0228	(0022)
0228	21 0070 0073	(0070) = 69 (Adke+Nbe+Nbt+p-1) 0003 en position facteur, nous avons l'adresse augmentée de un , du dernier second membre de l'équation pivot.
0073	60 0050 0117	(0050)
0117	Voir en (6) dans le Sous Programme d'Elimination.	

(7) 0075 SORTIE du Sous Programme d'Elimination.

Il faut distinguer deux cas.

1°) Nous avons Nbp = Nbt - 1

Il nous faut détecter le dernier terme du premier membre de l'équation pivot. Ainsi nous pourrons calculer et mettre en place le terme de rang (k+Nbp) de l'équation sur laquelle nous calculons.

0075	10 0155 0209	(0155) = oo (p) oooo
0209	10 0022 0029	(0022)
0029	44 0150 0034	(8003) = (0050) - [(0070)-(0155) -(0022)] = 69 xxxx 0003 - 69 [Adke+Nbe +Nbt - 2] 0003

Le rang du prochain terme de l'équation pivot à utiliser, n'est pas égal à (k + Nbt - 1), ou (k + Nbp).
Nous effectuons le calcul d'un terme.

(8) 0150 Sous-Programme d' ELIMINATION 0075

Nous avons à ce moment:
(0075) = (0008)

Voir page précédente en (7)

Le rang du prochain terme de l'équation pivot à utiliser est égal à (k + Nbt - 1) ou (k + Nbp).
Préparation du calcul du terme de rang (k+Nbp).

0034	69 0037 0040	(0037) = 44 0150 0042
0040	24 0075 0079	(0075) Modification de la SORTIE du Sous Programme d'Elimination, pour les calculs sur les seconds membres.
0079	65 0014 0120	(0014) = 69 xxxx 0053 en position facteur, nous avons l'adresse du premier second membre de l'une des (k-1) équations sur lesquelles porte la k° élimination.
0120	16 0022 0129	(0022) = oo oool oooo
0129	20 0014 0170	(0014) = 69 xxxx 0053 en position facteur, nous avons l'adresse d'une mémoire actuellement disponible. Nous allons y placer la valeur initiale du terme de rang (k + Nbp), c'est à dire zéro.
0170	69 0123 0076	(0123) = 21 xxxx 0150

```
0076    22 0123 8001        (0123) = 21 xxxx 0150
                            (8003) = oo oooo oooo
8001    21 xxxx 0150
0150                (8)
```

2°) Nous avons Nbp = Nbt

ou bien, nous travaillons sur les seconds membres de l'une des (k-1) équations sur lesquelles porte la k° élimination, dans le cas où : Nbp = Nbt - 1 .

```
0075    44 0150 0042
```

Tous les termes de l'équation sur laquelle nous travaillons, n'ont pas été utilisés.
Nous calculons avec le terme suivant.

```
0150                (8)
```

Tous les termes de l'équation sur laquelle nous travaillons, ont été utilisés.
Nous sommes prêts à calculer avec l'équation suivante.

```
0042    60 0038 0143        (0038) = comptage des équations
                                     utilisées.
0143    10 0022 0179        (0022) = oo oool oooo
0179    21 0038 0191        (0038)
0191    10 0094 0149        (0094) = - oo (k - 1) oooo
0149    44 0051 0208
```

La k° inconnue n'a pas été éliminéé dans les (k-1) premières équations.

```
0051                (5)
```

La k° inconnue a été éliminée dans les (k-1) premières équations.
La k° élimination est terminée; nous allons bloquer la k° équation (équation pivot)à la suite des (k-1) nouvelles équations que nous venons d'obtenir.

```
0208    65 0059 0214        (0059) = oo (Adke + Nbe) oooo
                            en adresse facteur, nous avons
                            l'adresse du premier terme de
                            l'équation pivot.
0214    15 0220 0126        (0220) = 69 oooo 8002
0126    20 0081 0084        (0081) = 69 (Adke + Nbe) 8002
0084    15 0036 0241        (0036) = oo (Nbt + p) oooo
0241    16 0022 0229        (0022) = oo oool oooo
0229    20 0070 0173        (0070) = 69 [Adke+Nbe+Nbt+p-1] 8002
                            en position facteur, nous avons
                            l'adresse, augmentée de l'unité,
                            du dernier second membre de
                            l'équation pivot.
0173    65 0064 0221        (0064) = 21 xxxx 0203
                            en position facteur, nous avons
                            l'adresse, augmentée de l'unité,
                            du dernier résultat enregistré.
0221    69 0024 0130        (0024) = 24 oooo 0060
0130    22 0024 0180
0180    65 8001 0039
0039    10 0081 8001        (0081)
(9) 8001    69 xxxx 8002
8002    24 xxxx 0060
0060    10 0022 0230        (0022) = oo oool oooo
0230    15 8001 0089
0089    11 0070 0176        (0070)
0176    44 0131 0181
```

Tous les termes de la k° équation (équation pivot) n'ont pas été déplacés.

```
0131    10 8001 8003    (9)
```

Le blocage de la k° équation est terminé.

```
0181    SORTIE du Sous Programme de la k° Elimination.
        Nous retournons dans le Programme de LIAISON.
```

ANNEXE II

Systèmes linéaires

Nous trouverons ici la liste des données et des résultats exacts des principaux systèmes linéaires auxquels nous nous sommes référés au cours de la première partie.

Tous ces systèmes linéaires sont présentés de la même façon. Dans le titre, nous donnons les valeurs de n (ordre du système), de p (nombre de séries de seconds membres) et éventuellement de m (nombre de diagonales constituant la bande) dans le cas particulier d'un système linéaire à MATRICE BANDE. Ensuite nous avons les données du système linéaire, listées équation par équation. Chaque équation comprend le premier et seconds membres. Finalement nous trouvons les RESULTATS exacts.

Modèle de ligne utilisé pour les Données

- Mot 1 :

+

avec «NE» : numéro de l'équation. Nous aurons $1 \leq NE \leq n$

«NL» : numéro de ligne pour l'équation. Nous aurons : $1 \leq NL \leq \frac{n+p}{7} + (0 \text{ ou } 1)$

«NB» : nombre de mots de la ligne. $1 \leq NB \leq 7$

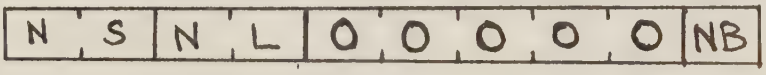
- Mots 2 à 8 :

Ils contiennent les coefficients de l'équation exprimés en Virgule Flottante du type *Flair*.

Modèle de ligne utilisé pour les Résultats

Chaque série de racines (ou vecteur solution) est listé suivant le modèle de ligne ci-dessous :

- Mot 1 :



avec «NS» : numéro d'ordre du vecteur solution. Nous aurons : $1 \leq NS \leq p$

«NL» : numéro de ligne pour le vecteur solution. Nous aurons : $1 \leq NL \leq \frac{n}{7} + (0 \text{ ou } 1)$

«NB» : nombre de mots de la ligne $1 \leq NB \leq 7$

- Mots 2 à 8 : Il contiennent les valeurs exactes des racines exprimées en Virgule Flottante du type *Flair*.

Remarque

Pour les systèmes linéaires à MATRICE BANDE nous avons adopté la même présentation que pour les systèmes quelconques. Toutefois il faut noter que pour chaque équation nous ne donnons que les coefficients de la bande, à priori non nuls, et les seconds membres. Dans ce cas particulier, nous aurons donc au maximum (m + p) coefficients par équation.

II-1 SYSTEME LINEAIRE N = 6 P = 1

DONNEES

01010000007	51210100000	50595000000	48800000000	49750000000	50347000000	52282700000	52351073000
02010000007	52237200000	50631000000	-50932000000	50714000000	-50201000000	51210100000	51580330000
03010000007	50631000000	52189100000	50938000000	-50174000000	50616000000	50595000000	52231317000
04010000007	-50932000000	50938000000	51995000000	50871000000	50201000000	48800000000	52245835000
05010000007	50714000000	-50174000000	50871000000	52112800000	50997000000	49750000000	52369665000
06010000007	-50201000000	50616000000	50201000000	50997000000	-52315500000	50347000000	-51172330000

RESULTATS

-01010000006	50100000000	51110000000	51210000000	51310000000	50200000000	51120000000

II-2 SYSTEME LINEAIRE N = 12 P = 1

DONNEES

01010000007	50554000000	51101100000	-50555000000	48800000000	50463000000	-50432000000	-503940 0000
01020000006	49750000000	50782000000	-50331000000	-51257400000	52189700000	52554593 00	
02010000007	52237200000	50631000000	-50932000000	50714000000	-50201000000	51210100000	-497100 0000
02020000006	-50749000000	-51137100000	51100100000	50817000000	50554000000	51784760000	
03010000007	50631000000	52189100000	50938000000	-50174000000	50616000000	50595000000	503780 0000
03020000006	-50947000000	49920000000	-50631000000	-50333000000	51101100000	52227106 00	
04010000007	-50932000000	50938000000	51995000000	50871000000	50201000000	48800000000	-501170 0000
04020000006	-50322000000	50711000000	49130000000	49110000000	-50555000000	52217197 00	
05010000007	50714000000	-50174000000	50871000000	-52112800000	50997000000	49750000000	-502330 0000
05020000006	-50444000000	48100000000	-50412000000	-50210000000	48800000000	52340412 00	
06010000007	-50201000000	50616000000	50201000000	50997000000	-52315500000	50347000000	507130 0000
06020000006	50131000000	50313000000	50293000000	50708000000	50463000000	51389560 00	
07010000007	51210100000	50595000000	48800000000	49750000000	50347000000	52282700 00	-506640 0000
07020000006	50202000000	49910000000	-50100000000	-50531000000	-50432000000	52315433 00	
08010000007	-49710000000	50378000000	-50117000000	-50233000000	50713000000	-50664000000	521703 0000
08020000006	-51103300000	50977000000	-50716000000	49190000000	-50394000000	52310527 00	
09010000007	-50749000000	-50947000000	-50322000000	-50444000000	50131000000	50202000000	-511033 0000
09020000006	-51933000000	-50272000000	50198000000	-50407000000	49750000000	-52355420 00	
10010000007	-51137100000	49920000000	50711000000	48100000000	50313000000	49910000000	509770 0000
10020000006	-50272000000	-52387700000	51177900000	50981000000	50782000000	-51157030 00	
11010000007	51100100000	-50631000000	49130000000	-50412000000	50293000000	-50100000000	-507160 0000
11020000006	50198000000	51177900000	52111200000	-50404000000	-50331000000	52101213 00	
12010000007	50817000000	-50333000000	49110000000	-50210000000	50708000000	-50531000000	491900 0000
12020000006	-50407000000	50981000000	-50404000000	52232500000	-51257400000	52420812 00	

RESULTATS

-01010000007	50100000000	51110000000	51210000000	51310000000	50200000000	51120000000	512200 0000
-01020000005	51320000000	50300000000	51130000000	51230000000	51330000000		

DONNEES

0101000007	-5066200000	5074400000	4977000000	5030200000	-5062100000	-5040300000	5010200000
0102000007	5017800000	5022600000	-5033500000	5040800000	-5043700000	5021700000	5077700000
0103000005	-5037000000	5036200000	-5120000000	5199100000	5216547300		
0201000007	5223720000	5063100000	-5093200000	5071400000	-5020100000	5121010000	-4971000000
0202000007	-5074900000	-5113710000	5110010000	5061700000	5055400000	5113120000	-4933000000
0203000005	5063800000	5031500000	5072700000	-5066200000	5210298900		
0301000007	5063100000	5218910000	5093800000	-5017400000	5061600000	5059500000	5037800000
0302000007	-5094700000	4992000000	-5063100000	-5033300000	5110110000	5070100000	4966000000
0303000005	5083900000	5043800000	5049400000	5074400000	5227949200		
0401000007	-5093200000	5093800000	5192000000	5087100000	5020100000	4880000000	-5011700000
0402000007	-5032200000	5071100000	4913000000	4911000000	-5055500000	5047700000	4956000000
0403000005	5013300000	-5056100000	4919000000	4977000000	5220525700		
0501000007	5071400000	-5017400000	5087100000	5211280000	5099700000	4975000000	-5023300000
0502000007	-5044400000	4810000000	-5041200000	-5021000000	4880000000	5019800000	5034500000
0503000005	-5076900000	5052200000	-5031700000	5030200000	5234586700		
0601000007	-5020100000	5061600000	5020400000	5099700000	-5231550000	5034700000	5071300000
0602000007	5013100000	5031300000	5029300000	5070800000	5046300000	5021000000	-5081200000
0603000005	5053300000	5065000000	5042100000	-5062100000	5156110000		
0701000007	5121010000	5059500000	4880000000	4975000000	5034700000	5228270000	-5066400000
0702000007	5020200000	4991000000	-5010000000	-5053100000	-5043200000	5075600000	4964000000
0703000005	-5014400000	-5025600000	5093600000	-5040300000	5230582800		
0801000007	-4971000000	5037800000	-5011700000	-5023300000	5071300000	-5066400000	5217030000
0802000007	-5110330000	5097700000	-5071600000	4919000000	-5039400000	5042100000	-5031700000
0803000005	-5068100000	5011800000	-5052900000	5010200000	5229432600		
0901000007	-5074900000	-5094700000	-5032200000	-5044400000	5013100000	5020200000	-5110330000
0902000007	-5193300000	-5027200000	5019800000	-5040700000	4975000000	-4933000000	5022300000
0903000005	5041700000	5050100000	5025500000	5017800000	-5232144300		
1001000007	-5113710000	4992000000	5071100000	4810000000	5031300000	4991000000	5097700000
1002000007	-5027200000	-5236770000	5117790000	5098100000	5078200000	5031600000	4870900000
1003000005	5063600000	5019300000	5072400000	5022600000	5114495000		
1101000007	5110010000	-5063100000	4913000000	-5041200000	5029300000	-5010000000	-5071600000
1102000007	5019800000	5117790000	5211120000	-5040400000	-5033100000	5064200000	-5052700000
1103000005	5011200000	4880000000	4974000000	-5033300000	5194708000		
1201000007	5081700000	-5033200000	4911000000	-5021000000	5070800000	-5053100000	4919000000
1202000007	-5043700000	5098100000	-5040400000	5223250000	-5125740000	5112730000	-5061200000
1203000005	4670000000	-5030100000	-5027900000	5040800000	5241199500		
1301000007	5055400000	5110110000	-5055500000	4880000000	5046300000	-5043200000	-5039400000
1302000007	4975000000	5078200000	-5033100000	-5125740000	5218970000	-5117030000	5020700000
1303000005	-5015400000	-5034700000	5056100000	-5043700000	5253143500		
1401000007	5113120000	5070100000	5047700000	5019800000	5021000000	5075600000	5042100000
1402000007	-4933000000	5031600000	5064200000	5112730000	-5117930000	5221680000	-5124310000
1403000005	5071400000	5087500000	-5061800000	5021700000	5212498700		
1501000007	-4933000000	4966000000	4956000000	5034500000	-5081200000	4964000000	-5031700000

DONNEES (SUITE)

1502000007	5022300000	4870000000	-5052700000	-5061200000	5020700000	-5124310000	5218390000
1503000005	-5130060000	-5091400000	5037100000	5077700000	5215582100		
1601000007	5063800000	5083900000	5013300000	-5076500000	5053300000	-5014400000	-5068100000
1602000007	5041700000	5063600000	5011200000	4870000000	-5015400000	5071400000	-5130060000
1603000005	5222290000	5088200000	-5091800000	-5037000000	5250066800		
1701000007	5031500000	5043800000	-5056100000	5052200000	5065000000	-5025600000	5011800000
1702000007	5050100000	5019300000	4880000000	-5030100000	-5034700000	5087500000	-5091400000
1703000005	5088200000	5213340000	5021600000	5036200000	5248064100		
1801000007	5072700000	5049400000	4919000000	-5081700000	5042100000	5093600000	-5052900000
1802000007	5025500000	5072400000	4974000000	-5027900000	5056100000	-5061800000	5037100000
1803000005	-5091800000	5021600000	5238110000	-5120000000	5215364300		

RESULTATS

-0101000007	5010000000	5111000000	5121000000	5131000000	5020000000	5112000000	5122000000
-0102000007	5132000000	5030000000	5113000000	5123000000	5133000000	5040000000	5114000000
-0103000004	5124000000	5134000000	5050000000	5115000000			

II-4 SYSTEME LINEAIRE N = 36 P = 1

DONNEES

```
01010000007 -50403000000  50741000000  49110000000 -50107000000 -50301000000 -50459000000  50379000000
01020000007  50420000000  50471000000  50351000000 -50144000000 -50721000000  49320000000  50410000000
01030000007  50158000000 -50213000000  50118000000 -50103000000 -50257000000  50542000000  50353000000
01040000007  50100000000 -50287000000 -49880000000  50131000000 -50712000000  50103000000  50770000000
01050000007 -50176000000  49780000000 -50212000000  50303000000 -50231000000 -51121100000  50876000000
01060000002  52354700000  53106972700
02010000007  52237200000  50631000000 -50932000000  50714000000 -50201000000  51210100000 -49710000000
02020000007 -50749000000 -51137100000  51100100000  50817000000  50554000000  51131200000 -49330000000
02030000007  50638000000  50315000000  50727000000 -50662000000  50998000000 -51170700000 -50575000000
02040000007 -51111100000  50179000000  50292000000  50760000000  50678000000  50318000000 -49970000000
02050000007  50571000000 -51108800000 -50750000000  50913000000  49490000000 -50378000000  50293000000
02060000002 -50403000000  51686120000
03010000007  50631000000  52189100000  50938000000 -50174000000  50616000000  50595000000  50378000000
03020000007 -50947000000  49920000000 -50631000000 -50333000000  51101100000  50701000000  49660000000
03030000007  50839000000  50438000000  50494000000  50744000000  50339000000 -51122700000 -50866000000
03040000007  50709000000  49190000000 -50391000000  50274000000  50399000000  50808000000 -51120000000
03050000007  50992000000  50777000000 -50344000000  50497000000 -50831000000  50579000000 -50975000000
03060000002  50741000000  52288885000
04010000007 -50932000000  50938000000  51995000000  50871000000  50201000000  48800000000 -50117000000
04020000007 -50322000000  50711000000  49130000000  49110000000 -50555000000  50477000000  49560000000
04030000007  50133000000 -50561000000  49190000000  49770000000  50887000000  50215000000  48300000000
04040000007  49370000000  50111000000 -50616000000  50214000000  50331000000  50661000000  48900000000
04050000007  50192000000 -50314000000  50678000000  50271000000 -50828000000  50101000000  48700000000
04060000002  49110000000  52224704000
05010000007  50714000000 -50174000000  50871000000  52112800000  50997000000  49750000000 -50233000000
05020000007 -50444000000  48100000000 -50412000000 -50210000000  48800000000  50198000000  50345000000
05030000007 -50765000000  50522000000 -50817000000  50302000000  48200000000 -50113000000  50213000000
05040000007  49270000000  49880000000  49150000000  50321000000 -50676000000  50387000000  50717000000
05050000007  50673000000  49710000000 -50806000000 -50901000000  50342000000 -50701000000  49720000000
05060000002 -50107000000  52344307000
06010000007 -50201000000  50616000000  50201000000  50997000000 -52315500000  50347000000  50713000000
06020000007  50131000000  50313000000  50293000000  50708000000  50463000000  50210000000 -50812000000
06030000007  50533000000  50650000000  50421000000 -50621000000  48500000000  50177000000  50349000000
06040000007 -50811000000 -50333000000  48600000000  50197000000  49980000000 -50610000000  50379000000
06050000007 -50421000000  50642000000  49270000000  50416000000  50193000000  50371000000  49440000000
06060000002 -50301000000  51524150000
07010000007  51210100000  50595000000  48800000000  49750000000  50347000000  52282700000 -50664000000
07020000007  50202000000  49910000000 -50100000000 -50531000000 -50432000000  50756000000  49640000000
07030000007 -50144000000 -50256000000  50936000000 -50403000000  50193000000 -50302000000 -50617000000
07040000007  50541000000 -50310000000 -50187000000  50279000000 -50341000000 -50654000000  49290000000
07050000007  50137000000 -50443000000  50177000000  50309000000 -50299000000  50643000000 -50375000000
07060000002 -50459000000  52242990000
08010000007 -49710000000  50378000000 -50117000000 -50233000000  50713000000 -50664000000  52170300000
08020000007 -51103300000  50977000000 -50716000000  49190000000 -50394000000  50421000000 -50317000000
08030000007 -50681000000  50118000000 -50529000000  50102000000  49770000000 -50279000000  50415000000
08040000007 -50703000000  50217000000  49880000000 -50444000000  50123000000  50345000000 -50121000000
```

DONNEES (SUITE)

0805000007	49690000000	50901000000	49550000000	-50267000000	50317000000	50521000000	-50482000000
0806000002	50379000000	52331893000					
0901000007	-50749000000	-50947000000	-50322000000	-50444000000	50131000000	50202000000	-51103300000
0902000007	-51933000000	-50272000000	50198000000	-50407000000	49750000000	-49330000000	50223000000
0903000007	50417000000	50501000000	50255000000	50178000000	-50257000000	50334000000	50432000000
0904000007	50171000000	50602000000	50597000000	50323000000	50272000000	50118000000	50258000000
0905000007	-49710000000	50515000000	50321000000	50408000000	50152000000	-50264000000	50207000000
0906000002	50420000000	-52219280000					
1001000007	-51137100000	49920000000	50711000000	48100000000	50313000000	49910000000	50977000000
1002000007	-50272000000	-52387700000	51177900000	50981000000	50782000000	50316000000	48700000000
1003000007	50636000000	50193000000	50724000000	50226000000	50342000000	50479000000	50251000000
1004000007	50292000000	50512000000	-50237000000	50118000000	50310000000	50405000000	50612000000
1005000007	50589000000	-50222000000	50419000000	50579000000	-50376000000	50197000000	50353000000
1006000002	50471000000	52104109000					
1101000007	51100100000	-50631000000	49130000000	-50412000000	50293000000	-50100000000	-50716000000
1102000007	50198000000	51177900000	52111200000	-50404000000	-50331000000	50642000000	-50527000000
1103000007	50112000000	48800000000	49740000000	-50335000000	50437000000	50124000000	49620000000
1104000007	-50178000000	50414000000	-50271000000	50349000000	50190000000	-49330000000	50172000000
1105000007	-50254000000	50378000000	-50531000000	50211000000	49550000000	50233000000	-50433000000
1106000002	50351000000	52125700000					
1201000007	50817000000	-50333000000	49110000000	-50210000000	50708000000	-50531000000	49190000000
1202000007	-50407000000	50981000000	-50404000000	52232500000	-51257400000	51127300000	-50612000000
1203000007	48700000000	-50301000000	-50279000000	50408000000	50177000000	-50429000000	49740000000
1204000007	-50707000000	-50106000000	50211000000	50282000000	-50355000000	50619000000	-50113000000
1205000007	50231000000	-50420000000	50334000000	-50225000000	50192000000	49130000000	-50228000000
1206000002	-50144000000	52395072000					
1301000007	50554000000	51101100000	-50555000000	48800000000	50463000000	-50432000000	-50394000000
1302000007	49750000000	50782000000	-50331000000	-51257400000	52189700000	-51170300000	50207000000
1303000007	-50153400000	-50347000000	50561000000	-50437000000	50122000000	-49430000000	-50135000000
1304000007	50368000000	50649000000	-50171000000	50379000000	50414000000	-50569000000	50722000000
1305000007	-50301000000	50216000000	50138000000	49170000000	49390000000	-50238000000	50124000000
1306000002	-50721000000	52530996000					
1401000007	51131200000	50701000000	50477000000	50198000000	50210000000	50756000000	50421000000
1402000007	-49330000000	50316000000	50642000000	51127300000	-51170300000	52216800000	-51243100000
1403000007	50714000000	50875000000	-50618000000	50217000000	49700000000	50137000000	50414000000
1404000007	-50522000000	-50669000000	50400000000	48800000000	50171000000	50331000000	50529000000
1405000007	50419000000	-50200000000	50303000000	50411000000	-50207000000	48200000000	50101000000
1406000002	49320000000	52148277000					
1501000007	-49330000000	49660000000	49560000000	50345000000	-50812000000	49640000000	-50317000000
1502000007	50223000000	48700000000	-50527000000	-50612000000	50207000000	-51243100000	52183900000
1503000007	-51300600000	-50914000000	50371000000	50777000000	-50432000000	-50173000000	50525000000
1504000007	50761000000	50882000000	-50311000000	50228000000	49220000000	50705000000	-50829000000
1505000007	-50602000000	48900000000	50198000000	-50202000000	50476000000	50600000000	-50302000000
1506000002	50410000000	52193825000					
1601000007	50638000000	50839000000	50133000000	-50765000000	50533000000	-50144000000	-50681000000
1602000007	50417000000	50636000000	50112000000	48700000000	-50154000000	50714000000	-51300600000
1603000007	52222900000	50882000000	-50918000000	-50370000000	-50479000000	50271000000	49270000000
1604000007	-50454000000	50731000000	49110000000	-50438000000	-50225000000	-50666000000	50120000000
1605000007	50145000000	-50405000000	50238000000	49730000000	-50500000000	50311000000	50229000000
1606000002	50158000000	52479294000					
1701000007	50315000000	50438000000	-50561000000	50522000000	50650000000	-50256000000	50118000000

DONNEES (SUITE)

17020000007	505c1000000	50193000000	48800000000	-50301000000	-50347000000	50875000000	-50914000000
17030000007	50882000000	52133400000	50216000000	50362000000	50254000000	50414000000	-50308000000
17040000007	50544000000	-50272000000	50150000000	49880000000	50112000000	50361000000	-50421000000
17050000007	50344000000	-50510000000	-50400000000	50128000000	50253000000	-50612000000	49170000000
17060000002	-50213000000	52501475000					
13010000007	50727000000	50494000000	49190000000	-50817000000	50421000000	50936000000	-50529000000
15020000007	50255000000	5.072400000	49740000000	-50279000000	50561000000	-50618000000	50371000000
18030000007	-50918000000	50216000000	52381100000	-51200000000	50604000000	-50200000000	48500000000
16040000007	-50311000000	50442000000	-50317000000	-50216000000	50335000000	-50502000000	50612000000
18050000007	-50311000000	50251000000	50466000000	-50111000000	48600000000	50319000000	-50222000000
15060000002	50118000000	52158381000					
19010000007	-50662000000	50744000000	49770000000	50302000000	-50621000000	-50403000000	50102000000
19020000007	50178000000	50226000000	-50335000000	50408000000	-50437000000	50217000000	50777000000
17030000007	-50370000000	50362000000	-51200000000	51991000000	-50123000000	50132000000	-50231000000
18040000007	50334000000	-50411000000	50326000000	-49180000000	-50113000000	50235000000	50415000000
19050000007	49980000000	50169000000	-50260000000	-50310000000	50137000000	-50231000000	48600000000
19060000002	-50103000000	52176603000					
20010000007	50998000000	50339000000	50887000000	48200000000	48500000000	50193000000	49770000000
20020000007	-50257000000	50342000000	50437000000	50177000000	50122000000	49700000000	-50432000000
20030000007	-50479000000	50254000000	50604000000	-50123000000	52161700000	-50971000000	50633000000
20040000007	-50716000000	-50829000000	50130000000	49970000000	50232000000	-50438000000	48500000000
20050000007	50211000000	-50321000000	49390000000	50172000000	-50901000000	50335000000	-50122000000
20060000002	-50257000000	52330633000					
21010000007	-51170700000	-51122700000	50215000000	-50113000000	50177000000	-50302000000	-50279000000
21020000007	50334000000	50479000000	50124000000	-50429000000	-49430000000	50137000000	-50173000000
21030000007	50271000000	50414000000	-50200000000	50132000000	-50971000000	52291000000	-51200000000
21040000007	-50999000000	50488000000	-50812000000	-50715000000	50353000000	-50230000000	-50178000000
21050000007	50339000000	50106000000	-50218000000	50822000000	48300000000	-50213000000	50302000000
21060000002	50542000000	52984579000					
21010000007	-50575000000	-50866000000	48300000000	50213000000	50349000000	-50617000000	50415000000
21020000007	50432000000	50251000000	49620000000	49740000000	-50135000000	50414000000	50525000000
21030000007	49270000000	-50308000000	48500000000	-50231000000	50633000000	-51200000000	52150200000
21040000007	50223000000	50312000000	50451000000	50505000000	50202000000	50413000000	50222000000
21050000007	50170000000	50340000000	50268000000	50452000000	50300000000	50502000000	50193000000
21060000002	50353000000	52142868000					
21010000007	-51111110000	50709000000	49370000000	49270000000	-50811000000	50541000000	-50703000000
23020000007	50171000000	50292000000	-50178000000	-50707000000	50388000000	-50522000000	50761000000
23030000007	-50454000000	50544000000	-50311000000	50334000000	-50716000000	-50999000000	50223000000
23040000007	52306100000	-51477500000	-51131200000	-50916000000	-50866000000	50451000000	49350000000
23050000007	-50152000000	-50662000000	50376000000	-50556000000	-50479000000	50116000000	-50209000000
23060000002	50100000000	52233715000					
24010000007	50179000000	49190000000	50111000000	49880000000	-50333000000	-50310000000	50217000000
24020000007	50602000000	50512000000	50414000000	-50106000000	50649000000	-50669000000	50882000000
24030000007	50731000000	-50272000000	50442000000	-50411000000	-50829000000	50488000000	50312000000
24040000007	-51477500000	52200100000	-50936000000	50411000000	50221000000	-50347000000	50514000000
24050000007	49200000000	50715000000	-50200000000	48500000000	-50139000000	50827000000	-50314000000
24060000002	-50287000000	52482801000					
25010000007	50292000000	-50391000000	-50616000000	49150000000	48600000000	-50187000000	49880000000
25020000007	50597000000	-50237000000	-50271000000	50211000000	-50171000000	50400000000	-50311000000
25030000007	49110000000	50150000000	-50317000000	50326000000	50130000000	-50812000000	50451000000
25040000007	-51131200000	-50936000000	52108600000	-50355000000	49110000000	-50236000000	-50414000000

DONNEES (SUITE)

2505000007	5011600000	5037900000	5044700000	-5015600000	4976000000	-4991000000	5012100000
2506000002	-4988000000	5232923800					
2601000007	5076000000	5027400000	5021400000	5032100000	5019700000	5027900000	-5044400000
2602000007	5032300000	5011800000	5034900000	5028200000	5037900000	4880000000	5022800000
2603000007	-5043800000	4988000000	-5021600000	-4918000000	4997000000	-5071500000	5050500000
2604000007	-5091600000	5041100000	-5035500000	5239130000	5099800000	-5070100000	5112440000
2605000007	-5034900000	-5077700000	5081200000	-5110040000	-5076600000	5053300000	-5023100000
2606000002	5013100000	5222916800					
2701000007	5067800000	5039900000	5033100000	-5067600000	4998000000	-5034100000	5012300000
2702000007	5027200000	5031000000	5019000000	-5035500000	5041400000	5017100000	4922000000
2703000007	-5022500000	5011200000	5033500000	-5011300000	5023200000	5035300000	5020200000
2704000007	-5086600000	5022100000	4911000000	5099800000	5262310000	-5127140000	-5119270000
2705000007	4991000000	5022300000	-5047800000	5091800000	5033800000	-5027700000	5044400000
2706000002	-5071200000	5310116420					
2801000007	5031800000	5080800000	5066100000	5038700000	-5061000000	-5065400000	5034500000
2802000007	5011800000	5040500000	-4933000000	5061900000	-5056900000	5033100000	5070500000
2803000007	-5066600000	5036100000	-5050200000	5023500000	-5043800000	-5023000000	5041300000
2804000007	5045100000	-5034700000	-5023600000	-5070100000	-5127140000	5217770000	5074300000
2805000007	-5066800000	-5011700000	4972000000	-5037600000	-5029300000	5071500000	5083300000
2806000002	5010300000	5244659900					
2901000007	-4997000000	-5112000000	4890000000	5071700000	5037900000	4929000000	-5012100000
2902000007	5025800000	5061200000	5017200000	-5011300000	5072200000	5052900000	-5082900000
2903000007	5012000000	-5042100000	5061200000	5041500000	4850000000	-5017800000	5022200000
2904000007	4935000000	5051400000	-5041400000	5112440000	-5119270000	5074300000	5239870000
2905000007	-5129370000	-5111200000	5019200000	5040700000	5052100000	5088900000	5064200000
2906000002	5077000000	5232387000					
3001000007	5057100000	5099200000	5019200000	5067300000	-5042100000	5013700000	4969000000
3002000007	-4971000000	5058900000	-5025400000	5023100000	-5030100000	5041900000	-5060200000
3003000007	5014500000	5034400000	-5031100000	4998000000	5021100000	5033900000	5017000000
3004000007	-5015200000	4920000000	5011600000	-5034900000	4991000000	-5066800000	-5129370000
3005000007	5220350000	5012700000	5023500000	5011100000	-5071200000	-5090100000	-5034700000
3006000002	-5017600000	5234880800					
3101000007	-5110880000	5077700000	-5031400000	4971000000	5064200000	-5044300000	5090100000
3102000007	5051500000	-5022200000	5037800000	-5042000000	5021600000	-5020000000	4890000000
3103000007	-5040500000	-5051000000	5025100000	5016900000	-5032100000	5010600000	5034000000
3104000007	-5066200000	5071500000	5037900000	-5077700000	5022300000	-5011700000	-5111200000
3105000007	5012700000	5240030000	-5130010000	-5111270000	5045600000	5070000000	4990000000
3106000002	4978000000	5311177760					
3201000007	-5075000000	-5034400000	5067800000	-5080600000	4927000000	5017700000	4955000000
3202000007	5032100000	5041900000	-5053100000	5033400000	5013800000	5030300000	5019800000
3203000007	5023800000	-5040000000	5046600000	-5026000000	4939000000	-5021800000	5026800000
3204000007	5037600000	-5020000000	5044700000	5081200000	-5047800000	4972000000	5019200000
3205000007	5023500000	-5130010000	5238820000	5098700000	-5052900000	5043300000	-5017100000
3206000002	-5021200000	5227993700					
3301000007	5091300000	5049700000	5027100000	-5090100000	5041600000	5030900000	-5026700000
3302000007	5040800000	5057900000	5021100000	-5022500000	4917000000	5041100000	-5020200000
3303000007	4973000000	5012800000	-5011100000	-5031000000	5017200000	5082200000	5045200000
3304000007	-5055600000	4850000000	-5015600000	-5110040000	5091800000	-5037600000	5040700000
3305000007	5011100000	-5111270000	5098700000	5212910000	4911000000	5020700000	5041300000
3306000002	5030300000	5226287900					
3401000007	4949000000	-5083100000	-5082800000	5034200000	5019300000	-5029900000	5031700000

DONNEES (SUITE)

3402000007	5015200000	-5037600000	4955000000	5019200000	4939000000	-5020700000	5047600000
3403000007	-5050000000	5025300000	4860000000	5013700000	-5090100000	4830000000	5030000000
3404000007	-5047900000	-5013900000	4976000000	-5078600000	5033800000	-5029300000	5052100000
3405000007	-5071200000	5045600000	-5052900000	4911000000	5220700000	-5091500000	5012200000
3406000002	-5023100000	5255211000					
3501000007	-5037800000	5057900000	5010100000	-5070100000	5037100000	5064300000	5052100000
3502000007	-5026400000	5.019700000	5023300000	4913000000	-5023800000	4820000000	5060000000
3503000007	5031100000	-5061200000	5031900000	-5023100000	5033500000	-5021300000	5050200000
3504000007	5011600000	5082700000	-4991000000	5055300000	-5027700000	5071500000	5088900000
3505000007	-5090100000	5070000000	5043300000	5020700000	-5091500000	5243030000	-5124880000
3506000002	-5112110000	5236559200					
3601000007	5029300000	-5097500000	4870000000	4972000000	4944000000	-5037500000	-5048200000
3602000007	5020700000	5035300000	-5043300000	-5022800000	5012400000	5010100000	-5030200000
3603000007	5022900000	4917000000	-5022200000	4860000000	-5012200000	5030200000	5019300000
3604000007	-5020900000	-5031400000	5012100000	-5023100000	5044400000	5083300000	5064200000
3605000007	-5034700000	4990000000	-5017100000	5041300000	5012200000	-5124880000	5220220000
3606000002	5087600000	5242606700					

RESULTATS

-0101000007	5010000000	5111000000	5121000000	5131000000	5020000000	5112000000	5122000000
-0102000007	5132000000	5030000000	5113000000	5123000000	5133000000	5040000000	5114000000
-0103000007	5124000000	5134000000	5050000000	5115000000	5125000000	5135000000	5060000000
-0104000007	5116000000	5126000000	5136000000	5070000000	5117000000	5127000000	5080000000
-0105000007	5118000000	5128000000	5090000000	5119000000	5129000000	5110000000	5120000000
-0106000001	5130000000						

II=5 SYSTEME LINEAIRE N= 6 P= 2

DONNEES

01010000007	52400000000	5275326800	5232733250	5261469697	5314715127	5226883094	5220000000
01020000001	5177031940						
02010000007	5275326800	5314715127	5261469697	5311976116	5329716280	5250338047	5241758300
02020000001	5217706839						
03010000007	5232733250	5261469697	5226883094	5250338047	5311976116	5222157254	5210806850
03020000001	5079828900						
04010000007	5261469697	5311976116	5250338047	5297803641	5324125068	5241366062	5223741601
04020000001	5142266160						
05010000007	5314715127	5329716280	5311976116	5324125068	5361769712	5297803641	5289650180
05020000001	5240924914						
06010000007	5226883094	5250338047	5222157254	5241366062	5297803641	5218326475	5143436618
06020000001	-5138316383						

RESULTATS

-02010000006	51800000000	52120000000	52170000000	-52150000000	-50300000000	-52300000000

II-6 SYSTEME LINEAIRE N = 4 P = 1

EXTRAIT D'UNE COMMUNICATION DE M.PETKAR P.MATIC (ELECTRICAL ENGINEER, INSTITUTE BORIS KIDRIC, BELGRADE YUGOSLAVIA) AUX JOURNEES INTERNATIONALES DE CALCUL ANALOGIQUE (SEPTEMBRE 1955)

DONNEES

0101000005	5612173400	5616921700	5617662400	5616666200	5663423700
0201000005	5616921700	5623522200	5624550500	5623165300	5688159700
0301000005	5617662400	5624550500	5625642300	5624202900	5692058100
0401000005	5616666200	5623165300	5624202900	5622847400	5686881800

RESULTATS

-0101000004	5110000000	5110000000	5110000000	5110000000

II-7 SYSTEME LINEAIRE A MATRICE BANDE
N = 12 M = 7 P = 1

DONNEES

01010000005	5097500000	5067200000	5022100000	5053300000	5051140000		
02010000006	5010500000	5089100000	5075800000	5026200000	5039700000	5071940000	
03010000007	5052700000	5024300000	5079300000	5015500000	5024400000	5017300000	5062700000
04010000007	5010200000	5035600000	5053100000	5090100000	5081700000	5061600000	5060100000
04020000001	5117999000						
05010000007	5052900000	5037700000	5069100000	5090500000	5016100000	4998000000	5012700000
05020000001	5112146000						
06010000007	5029900000	5041800000	5023800000	5070300000	5053500000	5027000000	4941000000
06020000001	5114251000						
07010000007	-5029700000	5051400000	5018900000	-5063700000	5051600000	5048700000	-4979000000
07020000001	5057780000						
08010000007	5040300000	5052000000	-5033200000	5082700000	-5050900000	-5013300000	5020100000
08020000001	5057270000						
09010000007	5013700000	-5045800000	-5038900000	5060700000	5051900000	-4997000000	5017100000
09020000001	5061420000						
10010000007	5029200000	5011000000	-5035700000	5071100000	-5053800000	5021900000	5035310000
11010000006	-5047500000	5029300000	5038200000	-5057400000	5016300000	-5017010000	
12010000005	5019500000	-5043900000	5055400000	5070700000	5111943000		

RESULTATS

01010000007	5010000000	5020000000	5030000000	5040000000	5050000000	5060000000	5070000000
01020000005	5080000000	5090000000	5110000000	5111000000	5112000000		

ANNEXE III

Bibliographie

SOMMAIRE

OUVRAGES GENERAUX

[1] HARTREE — Numerical Analysis - *Oxford University Press, 1955.*

[2] HILDEBRAND — Introduction to numerical analysis - *Mc Graw Hill, 1956.*

[3] HOUSEHOLDER — Principles of Numerical Analysis - *Mc Graw Hill, 1953.*

[4] MILNE — Numerical Calculus - *Princeton University Press, 1949.*

OUVRAGES SPECIALISES

[5] BODEWIG — Matrix Calculus - *North Holland, 1959.*

[6] DWYER — Linear Computations - *Wiley, 1951.*

[7] FOX — Practical Solution of Linear Equations and Inversion of Matrices. - *(National Bureau of Standards Applied Mathematics Series. 39), 1954.*

[8] HESTENES & STIEFEL — Journal of Research of the National Bureau of Standards, *Vol. 49, N° 6, December 1952.*

[9] — Mathematical Methods for Digital Computers, Part II : Matrices and Linear Equations, *Wiley, 1960.*

[10] — Modern Computing Methods (Chap. 1-2) - Notes on Applied Science. 16, *National Physical Laboratory, 1957.*

÷ ÷ ÷

Pour une bibliographie beaucoup plus détaillée, nous renverrons à :

[11] — Simultaneous Linear Equations and the Determination of Eigenvalues. *(National Bureau of Standards Applied Mathematics Series. 29), 1953.*

÷ ÷ ÷

[12] H. BAHUREL — *Thèse* - Etude d'un langage symbolique à macro-instructions et analyse pour I. B. M. 650 Standard, *1963.*

DEUXIEME THESE

Propositions données par la Faculté

RESOLUTION NUMERIQUE des PROBLEMES aux LIMITES

RELATIFS aux EQUATIONS DIFFERENTIELLES

Vu,
Le Directeur de Thèse,
G. BRILLOUET

Vu,
Le Président du Jury,
G. BRILLOUET

Vu et approuvé,
NANTES, le 4 mars 1963
Le Doyen de la Faculté des Sciences,
G. BRILLOUET

Vu et permis d'imprimer,
NANTES, le 9 mars 1963
Le Recteur de l'Académie
de NANTES
Président du Conseil de l'Université,
M. SCHMITT